TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Школа Инженерная школа информационных технологий и робототехники
Направление подготовки 01.03.02 Прикладная математика и информатика
Отделение школы (НОЦ) Отделение информационных технологий

БАКАЛАВРСКАЯ РАБОТА

Тема работы
Разработка алгоритма для генерации текста на основе нейросетевой модели

УДК 004.421.5.032.26:004.738.5:339

Студент

Группа	ФИО	Подпись	Дата
8Б61	Карнаухов Владислав Андреевич		

Руководитель ВКР

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ	Саврасов Федор Васильевич			

КОНСУЛЬТАНТЫ ПО РАЗДЕЛАМ:

По разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОСГН ШБИП	Подопригора Игнат Валерьевич	к.э.н.		

По разделу «Социальная ответственность»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Ассистент ООД	Матвиенко Владимир Владиславович			

ДОПУСТИТЬ К ЗАЩИТЕ:

Руководитель ООП	ФИО	Ученая степень, звание	Подпись	Дата
Доцент	Шевелев Геннадий Ефимович	к.ф.-м.н.		

Томск – 2020 г.

ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ ОБУЧЕНИЯ

Код Результатов	Результат обучения (выпускник должен быть готов)	Требования ФГОС, Критерии АИОР
Профессиональные компетенции		
Р1	Применять глубокие математические и профессиональные знания для решения задач научно-исследовательской, проектной, производственной и технологической деятельности в области системного и прикладного программирования.	Требования ФГОС (ОК-11, 12, ПК3, 10), Критерий 5 АИОР (п. 5.2.1), согласованный с требованиями международных стандартов *EURACE* и *FEANI* Требования профессиональных стандартов Ассоциации предприятий компьютерных и информационных технологий Требования работодателей: ФГУП «РФЯЦ-ВНИИТФ им. академика Е.И. Забабахина», ООО НАЦ «Недра», ИХН СО РАН
Р2	Умение использовать знания по естественнонаучным дисциплинам при определении задач математического моделирования объектов и явлений в различных предметных областях	Требования ФГОС (ПК-3,9) Критерий 5 АИОР (п.5.2.3), согласованный с требованиями международных стандартов *EURACE* и *FEANI* Требования работодателей: ФГУП «РФЯЦ-ВНИИТФ им. академика Е.И. Забабахина», ООО «НАПО им. В.П. Чкалова», ИХН СО РАН
Р3	Демонстрировать понимание сущности и значения информации в развитии современного общества, владение основными методами, способами и средствами получения, хранения, переработки информации; использование для решения коммуникативных задач современных технических средств и информационных технологий.	Требования ФГОС (ОК-5, 11, 12,14,15, ПК-2, 6), Критерий 5 АИОР (п. 5.2.2), согласованный с требованиями международных стандартов *EUR-ACE* и *FEANI* Требования профессиональных стандартов Ассоциации предприятий компьютерных и информационных технологий Требования работодателей: ООО «Контек-софт», ОАО «Газпром переработка», ООО Нижневартовскэнергонефть».

P4	Выполнять инновационные проекты с применением глубоких профессиональных знаний и эффективных методов проектирования для достижения новых результатов, обеспечивающих конкурентные преимущества в условиях экономических, экологических, социальных и других ограничений.	Требования ФГОС (ОК-14, ПК- 7, 9,14), Критерий 5 АИОР (п. 5.2.4), согласованный с требованиями международных стандартов *EURACE* и *FEANI* Требования профессиональных стандартов Ассоциации предприятий компьютерных и информационных технологий. Требования работодателей: ООО «Контек-софт», ОАО «Газпром переработка», ИХН СО РАН.
P5	Демонстрировать знание о формах организации образовательной и научной деятельности в высших учебных заведениях, *иметь навыки* преподавательской работы.	Требования ФГОС (ОК-1, 10, 16, ПК-1, 14, 15), Критерий 5 АИОР (п. 5.2.1), согласованный с требованиями международных стандартов *EURACE* и *FEANI*
P6	Способность осуществлять организационно-управленческую и социально-ориентированную деятельность с соблюдением профессиональной этики	Требования ФГОС (ОК-5,13,16, ПК-11-13,16) Критерий 5 АИОР (п.5.2.12-13) согласованный с требованиями международных стандартов *EURACE* и *FEANI*
Универсальные компетенции		
P7	Активно владеть иностранным языком на уровне, позволяющем работать в интернациональной среде, включая разработку документации и представление результатов инновационной деятельности. Толерантность в восприятии социальных и культурных различий.	Требования ФГОС (ОК-2, 3,4, 7, ПК-8). Критерий 5 АИОР (п. 5.2.11), согласованный с требованиями международных стандартов *EURACE* и *FEANI* и Требования профессиональных стандартов Ассоциации предприятий компьютерных и информационных технологий

Р8	Эффективно работать индивидуально, в качестве члена и руководителя группы, состоящей из специалистов различных направлений и квалификаций, демонстрировать ответственность за результаты работы и готовность следовать корпоративной культуре организации	Требования ФГОС (ОК-1,4, 6, ПК-8,11,12), Критерий 5 АИОР (пп. 5.2.9,5.2.13), согласованный с требованиями международных стандартов *EUR-ACE* и *FEANI u* Требования профессиональных стандартов Ассоциации предприятий компьютерных и информационных технологий. Требования работодателей: ООО «Контек-софт», ОАО «Газпром переработка», ООО Нижневартовскэнергонефть».
Р9	Самостоятельно учиться и непрерывно повышать квалификацию в течение всего периода профессиональной деятельности. Способность к интеллектуальному, культурному, нравственному и профессиональному саморазвитию.	Требования ФГОС (ОК-8,9,16, ПК-5, 11), Критерий 5 АИОР (5.2.14), согласованный с требованиями международных стандартов *EURACE* и *FEANI.* Требования работодателей: Контек, ОАО «Газпром переработка», ООО Нижневартовскэнергонефть».

TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Школа Инженерная школа информационных технологий и робототехники
Направление подготовки 01.03.02 Прикладная математика и информатика
Отделение школы (НОЦ) Отделение информационных технологий

УТВЕРЖДАЮ:
Руководитель ООП

_____ _____ _____
(Подпись) (Дата) (Ф.И.О.)

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

В форме:

Бакалаврской работы

(бакалаврской работы, дипломного проекта/работы, магистерской диссертации)

Студенту:

Группа	**ФИО**
8Б61	Карнаухову Владиславу Андреевичу

Тема работы:

Разработка алгоритма для генерации текста на основе нейросетевой модели	
Утверждена приказом директора (дата, номер)	

Срок сдачи студентом выполненной работы:	

ТЕХНИЧЕСКОЕ ЗАДАНИЕ:

Исходные данные к работе *(наименование объекта исследования или проектирования; производительность или нагрузка; режим работы (непрерывный, периодический, циклический и т. д.); вид сырья или материал изделия; требования к продукту, изделию или процессу; особые требования к особенностям функционирования (эксплуатации) объекта или изделия в плане безопасности эксплуатации, влияния на окружающую среду, энергозатратам; экономический анализ и т. д.).*	Предметом исследования является задача обработки естественного языка, а именно лингвистическое моделирование. Разрабатываемый алгоритм должно уметь генерировать различные варианты семантически правильного текста на основе переданного ему начального текста.
Перечень подлежащих исследованию, проектированию и разработке вопросов *(аналитический обзор по литературным источникам с целью выяснения достижений мировой науки техники в рассматриваемой области; постановка задачи исследования, проектирования,*	Обзор литературы, обзор аналогичных проектов, выбор оптимальной нейросетевой модели, исследование возможных способов синонимизации текста, выбор инструментов для работы (языки программирования, библиотеки).

конструирования; содержание процедуры исследования, проектирования, конструирования; обсуждение результатов выполненной работы; наименование дополнительных разделов, подлежащих разработке; заключение по работе).	

Консультанты по разделам выпускной квалификационной работы	
Раздел	**Консультант**
Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	Подопригора Игнат Валерьевич, доцент ОСГН ШБИП
Социальная ответственность	Матвиенко Владимир Владиславович, ассистент ООД

Дата выдачи задания на выполнение выпускной квалификационной работы по линейному графику	

Задание выдал руководитель:

Должность	**ФИО**	**Ученая степень, звание**	**Подпись**	**Дата**
Доцент ОИТ	Саврасов Федор Витальевич	к.т.н.		

Задание принял к исполнению студент:

Группа	**ФИО**	**Подпись**	**Дата**
8Б61	Карнаухов Владислав Андреевич		

TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Школа Инженерная школа информационных технологий и робототехники
Направление подготовки 01.03.02 Прикладная математика и информатика
Отделение школы (НОЦ) Отделение информационных технологий

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«ФИНАНСОВЫЙ МЕНЕДЖМЕНТ, РЕСУРСОЭФФЕКТИВНОСТЬ И РЕСУРСОСБЕРЕЖЕНИЕ»

Студенту:

Группа	**ФИО**		
8Б61	Карнаухову Владиславу Андреевичу		
Школа	**ИШИТР**	**Отделение школы (НОЦ)**	**ОИТ**
Уровень образования	Бакалавр	**Направление/специальность**	01.03.02 Прикладная математика и информатика

Исходные данные к разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»:	
1. Стоимость ресурсов научного исследования (НИ): материально-технических, энергетических, финансовых, информационных и человеческих	1.Стоимость расходных материалов. 2.Стоимость расхода электроэнергии. 3.Норматив заработной платы.
2. Нормы и нормативы расходования ресурсов	1. Тариф на электроэнергию. 2. Коэффициенты для расчёта заработной платы.
3. Используемая система налогообложения, ставки налогов, отчислений, дисконтирования и кредитования	1. Отчисления во внебюджетные фонды (30%)
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. Оценка коммерческого потенциала, перспективности и альтернатив проведения НИ с позиции ресурсоэффективности и ресурсосбережения	1.Потенциальные потребители результатов исследования. 2.Анализ конкурентных технических решений. 3.SWOT – анализ.
2. Планирование и формирование бюджета научных исследований	1.Разработка структуры работы в рамках научного исследования; 2.Определение трудоемкости выполнения работ и разработка графика проведения научного исследования; 3. Бюджет научно – технического исследования.
3. Определение ресурсной (ресурсосберегающей), финансовой, бюджетной, социальной и экономической эффективности исследования	1. Определение показателей ресурсоэффективности разработки
Перечень графического материала (с точным указанием обязательных чертежей)**:**	
1. Оценка конкурентоспособности технических решений *2.* Матрица SWOT *3.* Альтернативы проведения НИ *4.* График проведения и бюджет НИ *5.* Оценка ресурсной, финансовой и экономической эффективности НИ	
Дата выдачи задания для раздела по линейному графику	

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОСГН ШБИП	Подопригора И.В.	к.э.н.		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8Б61	Карнаухов Владислав Андреевич		

TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Школа Инженерная школа информационных технологий и робототехники
Направление подготовки 01.03.02 Прикладная математика и информатика
Отделение школы (НОЦ) Отделение информационных технологий

ЗАДАНИЕ ДЛЯ РАЗДЕЛА «СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ»

Студенту:

Группа	ФИО
8Б61	Карнаухову Владиславу Андреевичу

Школа	ИШИТР	Отделение (НОЦ)	ОИТ
Уровень образования	Бакалавриат	**Направление/специальность**	01.03.02 Прикладная математика и информатика

Тема ВКР:

Разработка алгоритма для генерации текста на основе нейросетевой модели	
Исходные данные к разделу «Социальная ответственность»:	
1. Характеристика объекта исследования (вещество, материал, прибор, алгоритм, методика, рабочая зона) и области его применения	В работе разрабатывается алгоритм для генерации текста на основе нейросетевой модели. Область применения – SEO, интернет маркетинг.
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. Правовые и организационные вопросы обеспечения безопасности: – специальные (характерные при эксплуатации объекта исследования, проектируемой рабочей зоны) правовые нормы трудового законодательства; – организационные мероприятия при компоновке рабочей зоны.	Трудовой кодекс Российской Федерации от 30.12.2001 N 197-ФЗ (ред. от 27.12.2018); ГОСТ 12.2.032-78 ССБТ; ГОСТ 21889-76; ГОСТ 22269-76; ГОСТ Р 50923-96; СанПиН 2.2.2/2.4.1340-03; Федеральный закон от 22.08.1996 №125-ФЗ
2. Производственная безопасность: 2.1. Анализ выявленных вредных факторов 2.2. Анализ выявленных опасных факторов	Вредные и опасные факторы: - Недостаточная освещенность рабочей зоны; - Превышение уровня шума; - Отклонение показателей микроклимата; - Повышенное значение напряжения в электрической цепи.

3. Экологическая безопасность:	При эксплуатации ЭВМ потребляется электроэнергия, вырабатываемая на электростанциях, сопровождаемая выбросами различных вредных веществ в окружающую среду. Конструкция ЭВМ содержит различные пластиковые и металлические элементы, которые в случае прихода в негодность должны быть соответствующим образом утилизированы или переданы на вторичную обработку. Рассмотрены решения по обеспечению экологической безопасности.
4. Безопасность в чрезвычайных ситуациях:	– возможные ЧС – природные и техногенные, к которым можно отнести как сильный мороз, так и возможная диверсия; – типичная ЧС– пожар на рабочем месте.

Дата выдачи задания для раздела по линейному графику	

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Ассистент ООД	Матвиенко В. В.			

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8Б61	Карнаухов Владислав Андреевич		

TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Школа Инженерная школа информационных технологий и робототехники
Направление подготовки (специальность) 01.03.02 Прикладная математика и информатика
Уровень образования бакалавр
Отделение школы (НОЦ) Отделение информационных технологий
Период выполнения (осенний / весенний семестр 2019 /2020 учебного года)

Форма представления работы:

Бакалаврская работа

(бакалаврская работа, дипломный проект/работа, магистерская диссертация)

КАЛЕНДАРНЫЙ РЕЙТИНГ-ПЛАН
выполнения выпускной квалификационной работы

Срок сдачи студентом выполненной работы:	

Дата контроля	Название раздела (модуля) / вид работы (исследования)	Максимальный балл раздела (модуля)
	Написание основной части	75
	Написание части финансового менеджмента, ресурсоэффективности и ресурсосбережения	15
	Написание части социальной ответственности	10

СОСТАВИЛ:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ	Саврасов Федор Витальевич	к.т.н.		

СОГЛАСОВАНО:

Руководитель ООП

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Руководитель ООП	Шевелев Геннадий Ефимович	Доцент, к.ф. – м.н.		

РЕФЕРАТ

Выпускная квалификационная работа содержит 101 страницу, 17 рисунков, 21 таблицу, 3 приложения и 30 литературных источников.

Ключевые слова: лингвистическое моделирование, NLP, BERT, Transformer, синонимизация, генерация текста.

Объектом исследования являются нейросетевые лингвистические модели.

Цель работы: разработка алгоритма для генерации текста на основе нейросетевой модели.

В процессе исследования проводились: анализ предметной области, изучение лингвистических моделей, изучение BERT.

В процессе выполнения работы использовалась интегрированная среда разработки PyCharm для создания программной системы.

В результате был разработан алгоритм для генерации текста на основе статической лингвистической модели типа «Transformer».

В первой главе представлено описание предметной области, обзор традиционных лингвистических моделей.

Во второй главе представлен обзор современных нейросетевых моделей, используемых для задач обработки естественного языка.

В третьей главе представлена реализация алгоритма синонимизации.

В четвертой главе представлено выполненное задание по разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение», при выполнении которого были использованы данные анализа в области проектного и финансового менеджмента.

Пятая глава представляет собой выполненное задание по разделу «Социальная ответственность», где были рассмотрены аспекты производственной и экологической безопасности, безопасности в чрезвычайных ситуациях, а также правовые вопросы организации труда.

Содержание

Введение

Качество и разнообразие текстового контента на страницах сайтов имеет непосредственное влияние на позиционирование сайта в выдаче поисковыми системами и, соответственно, популярность ресурса. Писать разнообразный контент долго и трудоёмко, поэтому есть объективная необходимость в автоматизации данного процесса.

Современные нейросетевые модели показывают неплохие результаты в задачах обработки естественного языка. Одним из возможных способов синонимизации текста является использование статической лингвистической модели типа Transformer BERT. Её особенностью является возможность оценивать вероятность упоминания слова в контексте всего предложения. Например, для текста «…красный свитер» можно предсказать прилагательное «прекрасный». Необходимо разработать метод синонимизации текстов на основе данной модели.

Дополнительно в данной работе были выполнены задания по разделам «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение» и «Социальная ответственность», позволяющие оценить необходимость проведения данной работы и реальность внедрения результатов работы в реальную деятельность организаций.

1 Анализ предметной области

Этапы анализа текста на естественном языке практически не зависят от выбранного языка.

Обычно различают следующие этапы анализа текста:

- графематический анализ – выделение структурных единиц из входного текста;
- морфологический анализ – определение морфологических характеристик каждого слова – часть речи, падеж, склонение, спряжение и т.д.;
- синтаксический анализ – построение синтаксического представления предложения;
- семантический анализ – построение аргументно-предикатной структуры высказываний или другого вида семантического представления предложения.

На каждом из этих этапов возникают различного рода неоднозначности, и из-за невозможности кодирования всех явных и неявных знаний о языке, для решения многих задач обработки естественного языка используются статистические лингвистические модели [1].

Лингвистическое моделирование является фундаментальной задачей в обработке естественного языка (NLP) и понимании языка. Задача лингвистического моделирования заключается в том, чтобы назначить вероятность того, что встретится некоторое предложение или последовательность слов [2, 3, 4]. Это задача является фундаментальной для распознавания речи, оптического распознавания символов, исправления орфографии, машинного перевода. Помимо присвоения вероятности каждой последовательности слов, лингвистические модели также назначают вероятность того, что данное слово или последовательность слов будет следовать после заданной последовательности слов.

Считается, что качество решения задачи лингвистического моделирования идеально, если количество попыток предсказания следующего слова последовательности машиной меньше или равно количеству попыток, которое понадобилось бы человеку.

1.1 Традиционные лингвистические модели

Умение назначить вероятность появления слова за последовательностью слов $P(W_i|W_1, W_2, \dots, W_{i-1})$ и умение назначить вероятность некоторой последовательности слов $P(W_1, W_2, \dots, W_i)$ эквивалентны, так как одна выводится из другой. Умея моделировать вероятности последовательностей, можно записать условную вероятность слова в виде отношения вероятностей двух последовательностей:

$$P(W_i|W_1, W_2, \dots, W_{i-1}) = \frac{P(W_1, W_2, \dots, W_{i-1}, W_i)}{P(W_1, W_2, \dots, W_{i-1})}, \quad (1)$$

И наоборот, умея определять вероятность появления слова за последовательностью слов, можно воспользоваться цепным правилом для того, чтобы выразить вероятность некоторой последовательности как произведение условных вероятностей:

$$P(W_1, \dots, W_k) = P(W_1| < S >) \cdot P(W_2| < S >, W_1) \cdot \dots \times$$
$$\times P(W_k| < S >, W_1, \dots, W_{k-1}), \quad (2)$$

где <S> - специальный символ, обозначающий начало последовательности.

Из формулы (2) видно, что задача языкового моделирования сводится к последовательности задач предсказания слова, в которой каждое предсказание обусловлено предшествующими словами. Однако если рассматривать каждую последовательность слов в отдельности, то слишком часто будет встречаться такая последовательность слов, которая раньше не появлялась, и для которой нет предшествующих слов. В противном случае, будут попадаться последовательности, начало которых имеется в словаре, но продолжение

отсутствует. Для решения данной проблемы необходим способ группировки ранее встреченных последовательностей, который позволит предсказывать следующее слово. Один из возможных способов это марковское предположение.

Марковское предположение гласит, что будущее не зависит от прошлого при условии настоящего. Данное утверждение позволяет сказать, что следующее слово в последовательности зависит только от k последних слов, и тогда формула (1) приобретает вид:

$$P(W_{i+1}|W_{1:i}) \approx P(W_{i+1}|W_{i-k:i}), \quad (3)$$

Это позволяет построить n-граммную модель, в которой все встреченные последовательности имеют одинаковую длину n. Роль лингвистической модели в этом случае сводится к нахождению оценки $\hat{P}(W_{i+1}|W_{i-k:i})$ методом максимального правдоподобия:

$$\hat{P}(W_1, \ldots, W_n) = \frac{C(W_1, \ldots, W_n)}{N}, \quad (4)$$

$$\hat{P}(W_{i+1}|W_{i-k:i}) = \frac{C(W_{i-k:i+1})}{C(W_{i-k:i})}, \quad (5)$$

где $C(W_1, \ldots, W_n)$ – частота n-граммы в обучающемся тексте;

N – количество n-gram в словаре модели.

1.1.1 Ограничения традиционных лингвистических моделей

Традиционные методы лингвистического моделирования на основе метода максимального правдоподобия имеют следующие преимущества: простое обучение, масштабируемость на большие корпуса, также они показывают хорошие результаты на практике. Однако данные методы имеют несколько серьезных недостатков.

Масштабирование на более длинные n-граммы вызывает проблемы. В силу самой природы естественного языка и огромного количества слов в словаре, статистика для длинных n-грамм по необходимости будет

разреженной. Кроме этого, масштабирование на длинные *n*-граммы очень требовательно к памяти, зависимость количества возможных *n*-грамм от длинны *n*-граммы можно увидеть на таблице – 1 [3, 4].

Таблица 1 – Зависимость требуемой памяти от длинны *n*-граммы

Длинна *n*-граммы	Количество возможных *n*-gram для словаря размером 1000 слов.
3	10^9
4	10^{12}
5	10^{15}
6	10^{18}
7	10^{21}
8	10^{24}
9	10^{27}

Из таблицы 1 можно сказать, что количество наблюдаемых *n*-грамм кратно возрастает при увеличении длины *n*-граммы на 1. Исходя из этого, работать с более длинными моделями оказывается сложно из-за требований к памяти.

Помимо проблем с масштабируемостью, традиционные модели также страдают от недостатка обобщаемости на новые контексты, например, если имеются такие последовательности как «красный фломастер», «синий фломастер», то это никак не повлияет на оценку последовательности «зеленый фломастер», если она не встречалась прежде.

1.2 Нейросетевые лингвистические модели

Нейросетевые модели устраняют некоторые недостатки традиционных лингвистических моделей: они могут обрабатывать более длинные *n*-граммы ценой лишь линейного увеличения количества параметров, поддерживают обобщение контекста схожих слов [3].

Нейронная лингвистическая модель может быть разработана и использоваться автономно, например, для генерации новых последовательностей текста или быть основой многих других моделей, которые могут использоваться для таких задач, как: машинный перевод, вопрос-ответ, автоматическое распознавание речи.

Нейросетевые модели имеют один главный недостаток – необходимость обучения. Высокая производительность нейросетевой модели обусловлена долгим процессом обучения, который значительно дольше чем в традиционных моделях, и может занимать несколько неделей. Так как для этого требуется высокая производительность, то разработка нейросетевой модели значительно дороже традиционной, и поэтому для многих задач традиционные лингвистические модели на основе n-грамм все еще являются правильным выбором.

1.2.1 Структура нейросетевой модели

Рассмотрим принцип работы простой нейросетевой модели. Нейросетевая модель состоит из двух основных частей:

– Функция C, которая отвечает за векторное представление слов и представляет из себя распределенный вектор признаков, связанный с каждым словом в словаре V. На практике, C представляет собой входной словарь – матрицу признаков размерностью $|V| * m$, где m количество признаков для каждого слова. Размер словаря $|V|$ варьируется от 10000 до 1000000 слов, наиболее распространенным размеров является ~70000 слов.

– Функция активации g, которая принимает последовательность векторных представлений слов и определяет вероятность для каждого слова в словаре V быть следующим после рассматриваемого.

Таким образом, нейросетевую модель можно представить в виде следующей функции:

$$f(i, W_n, \dots, W_{n-k+1}) = g(i, C(W_n), \dots, C(W_{n-k+1})), \quad (6)$$

В большинстве случаев нейронная модель имеет один скрытый слой, расположенный за слоем с векторным представлением слов, который отвечает за распределение вероятностей для предсказуемого слова, для более удобного интерпретирования часто используется функция «softmax», которая преобразует вектор A размерности K в вектор B той же размерности, где каждый элемент B_i полученного вектора представляется вещественным числом в интервале [0,1] и $\sum_{i=1}^{n} B_i = 1$, функция «softmax» представлена в формуле 7:

$$softmax(A_i) = \frac{e^{A_i}}{\sum_{k=1}^{n} e^{A_k}}, \quad (7)$$

В конечном виде нейросетевую лингвистическую модель можно представить следующими уравнениями:

$$x = [C(W_1), C(W_2), \dots, C(W_i)], \quad (8)$$

$$h = g(x \times W + b), \quad (9)$$

$$z = h \times U, \quad (10)$$

$$\hat{y} = P(W_{i+1} | W_{i-k+1::i}) = softmax(g(W \times x + b) \times U), \quad (11)$$

где W – входная матрица весов размерностью $\mathrm{m} \times \mathrm{d_{hid}}$;

b – нейрон смещения;

h – результат функции активации, представляет собой вектор размерностью d_{hid}, d_{hid} – количество скрытых слоев;

U – выходная матрица весов размерностью $\mathrm{d}_{hid} \times |V|$

$\hat{y}$ – вектор размерностью $|V|$, где каждому слову из словаря V соответствует значение, отображающее с какой вероятностью оно может следовать за переданной последовательностью.

При переходе с n-граммы на n+1-грамму увеличивается размерность матрицы произведения $x \times W$ из формулы 9 с $n * m \times \mathrm{d_{hid}}$ до $(n+1) * m \times \mathrm{d}_{hid}$, небольшое увеличение числа параметров в противоположность

полиномиальному росту в случае традиционных, основанных на счетчиках моделей. Это возможно, потому что комбинации признаков вычисляются в скрытом слое. Увеличение n, скорее всего, потребует также увеличения количества скрытых слоев, однако это допустимый рост числа параметров по сравнению с традиционными моделями.

С каждым словарным словом ассоциирован один m–мерный вектор, получаемый в результате применения функции C к слову W, и один $\mathrm{d}_{\mathbf{hid}}$ – мерный вектор. Таким образом, добавление новых слов в словарь приводит к линейному увеличению количества параметров, что является положительным фактором, в отличие от традиционных моделей. Хотя во входном словаре производится только поиск, и его рост не оказывает существенного влияния на скорость вычислений, размер выходного вектора оказывает такое влияние, так как применение функции (7) к выходному слою подразумевает дорогостоящее умножение $h \times U$, за которым следует $|V|$ операций возведения в степень. Это вычисление определяет время выполнения, поэтому лингвистическое моделирование с большим размером словаря вызывает затруднения.

Препятствием для применения нейросетевых лингвистических моделей с большим выходным пространством может стать неприемлемо высокое время обучения и тестирования. Поиск эффективных методов работы с большими выходными пространствами – тема актуальных исследований. Ниже описаны некоторые из существующих решений [3]:

- иерархическая функция «softmax»;
- решение на основе самонормировки, например шумосопоставительное оценивание.

1.3 Перплексивность – оценка лингвистических моделей

Оценкой лингвистических моделей является перплексивность. Перплексивность – мера, показывающая, насколько хорошо статическая модель предсказывает выборку. Чем меньше перплексивность, тем лучше

аппроксимация. Если дан корпус текстов, содержащий n слов $W_1, W_2, \dots, W_{i-1}$, и функция лингвистической модели LM, назначающая вероятность слову на основе его истории, то перплексивность LM относительно корпуса равна:

$$PPL(LM) = 2^{-\frac{1}{n}\sum_{i=1}^{n} \log_2 LM(W_i|W_{1:i-1})}, \quad (12)$$

Хорошие языковые модели, отражающие реальное использование языка, назначают высокие вероятности события из корпуса, что характеризуется низкой перплексивностью.

Перплексивность – надёжный индикатор качества лингвистической модели. Перплексивность зависит от корпуса (перплексивность двух лингвистических моделей можно сравнить только относительно одного и того же корпуса).

1.4 Использование лингвистических моделей для генерации текста

Лингвистические модели можно использовать также для генерации предложений. После обучения модели на заданном наборе текстов возможно сгенерировать выборку случайных предложений из модели в соответствии с обученным ею распределением вероятностей. Для этого применяется следующий процесс: предсказать распределение вероятностей первого слова при условии начального символа и выбрать случайное слово в соответствии с этим распределением. Затем предсказать распределение вероятностей второго слова при условии первого, и так далее, пока не будет предсказан символ конца последовательности </S>. Уже при k=3 получается вполне приемлемый текст, а с увеличением k его качество растет.

При таком способе генерации текста на каждом шаге можно выбирать либо предсказанное слово с наивысшей оценкой, либо случайное слово из предсказанного распределения. Еще одним широко используемым вариантом для улучшения качества текста является использование лучевого поиска для нахождения предложения с наибольшей суммой вероятностей. Если брать на

каждом шаге предсказание с наивысшей вероятностью, то в результате иногда получается неоптимальная конечная вероятность, потому что процесс может зайти в тупик, выбрав слово, за которым следуют только слова с низкой вероятностью. Лучевой поиск работает следующим образом: на первом шаге из списка всех предложенных слов выбираем только N-слов с наивысшей вероятностью, на втором шаге для каждого слова из N-слов, полученных на первом шаге, так же выбираем N-наиболее вероятных слов, и так продолжаем до тех пор, пока не найдем конечное состояние.

2 Современные нейросетевые модели

На данный момент существует две ведущих архитектуры для лингвистического моделирования – рекуррентные нейронные сети (RNN) и трансформеры (Transformers) [5]. Первая обрабатывает входные токены один за другим для изучения отношений между ними, вторая получает последовательность токенов и изучает зависимости между ними, используя механизм внимания (attention).

Хотя обе архитектуры достигли впечатляющих достижений, их основным ограничением является получение долгосрочных зависимостей, например, использование важных слов в начале документа для прогнозирования слов в последующей части

2.1 Рекуррентные нейронные сети

Рекуррентные нейронные сети (RNN – Recurrent Neural Network) – класс моделей машинного обучения, основанный на использовании предыдущих состояний сети для вычисления текущего [6]. Такие сети удобно применять в тех случаях, когда входные данные задачи представляют собой нефиксированную последовательность значений (например, текстовые данные, где текстовый фрагмент представлен нефиксированным количеством предложений, фраз и слов). Каждый символ в тексте, отдельные слова, знаки препинания и даже целые фразы могут являться атомарным элементом входной последовательности. Во-первых, модели оценивают произвольные предложения на основе того, насколько часто они встречались в текстах. Это дает меру грамматической и семантической корректности. Такие модели используются в машинном переводе. Во-вторых, языковые модели генерируют новый текст. Обучение модели на поэмах Шекспира позволит в дальнейшем генерировать новый текст, похожий на его произведения.

Происходит это за счет того, что на каждом шаге обучения t значение скрытого слоя рекуррентной нейронной сети $h^t \in R^m$ вычисляется следующим образом:

$$h^t = f\ (W \times x^t + U \times h^{(t-1)} + b^h), \tag{13}$$

где $x^t \in R^n$ – входной вектор в момент времени t (например, векторное представление текущего слова в текстовом фрагменте);

$W \in R^{m \times n}$, $U \in R^{m \times m}$, $b^h \in R^m$ – обучаемые параметры рекуррентной нейронной сети;

f – функция нелинейного преобразования.

Чаще всего в качестве функции нелинейного преобразования выступает одна из следующих функций: сигмоидальная функция (14), гиперболический тангенс (15), «ReLu» (16).

$$f(x) = \sigma(x) = \frac{1}{1 + e^{(-x)}}, \tag{14}$$

$$f(x) = tanh(x) = \frac{e^{(x)} - e^{(-x)}}{e^{(x)} + e^{(-x)}}, \tag{15}$$

$$f(x) = max(0, x), \tag{16}$$

В простой рекуррентной нейронной сети (рисунок 1) выходное значение $y^t \in R^l$ на текущем шаге t вычисляется по формуле:

$$y^t = W \times h^t + b, \tag{17}$$

где $W \in R^{l \times m}$ и $b \in R^l$ – обучаемые параметры.

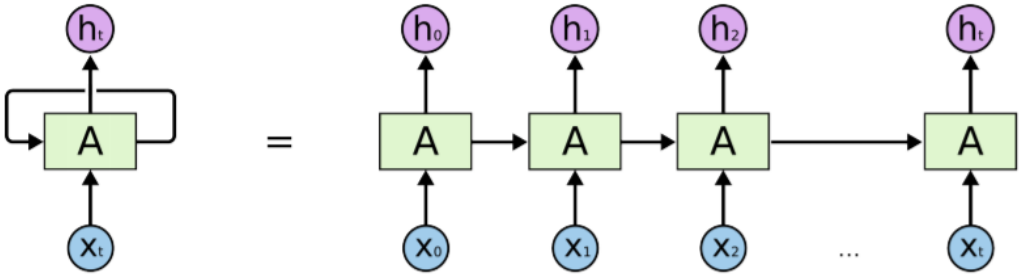


Рисунок 1 – Простейшая система рекуррентной сети

Применение данных систем используется во многих задачах обработки естественного языка. Стоит упомянуть, что наиболее часто используемым типом RNN являются LSTM (Long Short Term Memory), которые намного лучше захватывают долгосрочные зависимости, чем RNN. Более подробное рассмотрение рекуррентных сетей будет приведено в следующем подразделе. Вот некоторые примеры использования RNN в задачах обработки естественного языка:

- лингвистическое моделирование и генерация текстов;
- машинный перевод;
- распознавание речи;
- генерация описания изображений.

2.1.1 LSTM

В 1997 году Зепп Хохрайтер и Юрген Шмидхубер представили новый подход, получивший название LSTM (долгая краткосрочная память) [7]. Рекуррентные нейронные сети, основанные на этом подходе, имеют более сложный способ вычисления h^t. Данный способ, помимо входных значений и предыдущего состояния сети, использует также фильтры («gates»), определяющие, каким образом информация будет использоваться для вычисления как выходных значений на текущем слое y^t , так и значений скрытого слоя на следующем шаге h^{t+1}. Весь процесс вычисления h^t для простоты упоминается как LSTM-слой.

Все рекуррентные нейронные сети имеют форму цепочки повторяющихся модулей нейронной сети. В стандартных RNN этот повторяющийся модуль имеет простую структуру, например, один слой гиперболического тангенса, изображенного на рисунке 2**.**

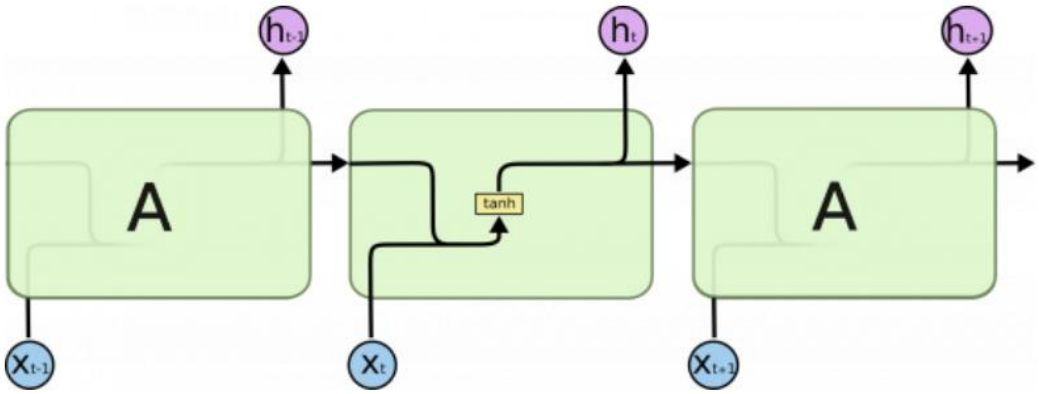


Рисунок 2 – Схематическое изображение повторяющегося модуля стандартной RNN, состоящей из одного слоя

В свою очередь на рисунке 3 изображена LSTM сеть, которая имеет явные отличия. Для того чтобы разобраться в них, рассмотрим их математические особенности.

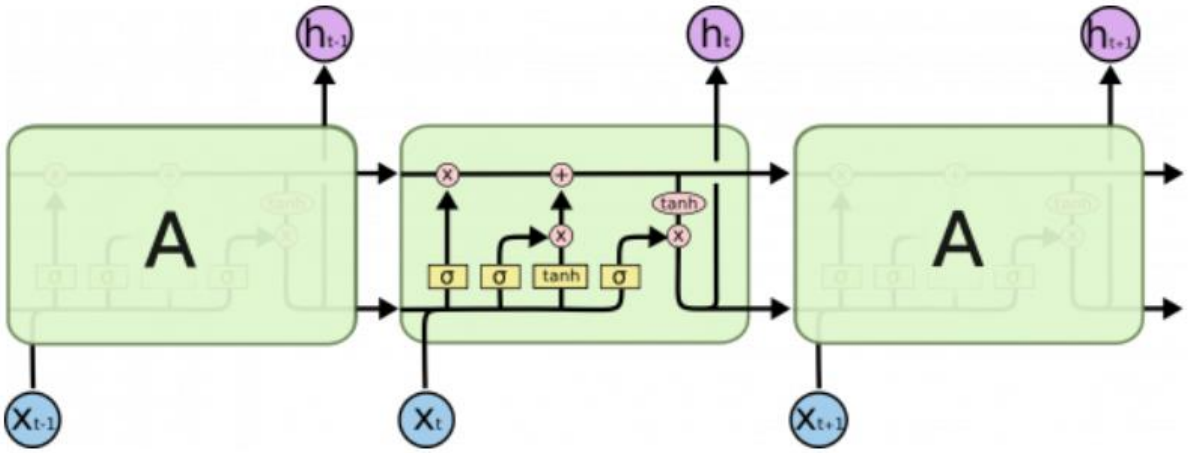


Рисунок 3 – Схематическое изображение повторяющегося модуля LSTM, состоящего из четырех взаимодействующих слоев

Даная модель разработана специально для устранения проблемы долгосрочной зависимости. Главной особенностью является запоминание информации в течение длительных периодов времени, что обеспечивает отсутствие необходимости в обучении.

Рассмотрим подробнее структуру LSTM-слоя. Центральным элементом здесь является запоминающий блок («memory cell»), который, наряду с состоянием сети *h*, вычисляется на каждом шаге, используя текущее входное

значение x, t и значение блока на предыдущем шаге c^{t-1}. Входной фильтр («input gate») i^t определяет, насколько значение блока памяти на текущем шаге должно влиять на результат. Значения фильтра варьируются от 0 (полностью игнорировать входные значения) до 1, что обеспечивается областью значений сигмоидальной функции:

$$i^t = \sigma(W^i x^t + U^i h^{t-1}), \quad (18)$$

Фильтр забывания («forget gate») позволяет исключить при вычислениях значения памяти предыдущего шага:

$$f^t = \sigma(W^f x^t + U^f h^{t-1}), \quad (19)$$

На основе всех данных, поступающих в момент времени t, вычисляется состояние блока памяти с t на текущем шаге, используя фильтры (20) и (21):

$$\check{c}^t = tanh\,(W x^t + U h^{t-1}), \quad (20)$$

$$c^t = f^t \cdot c^{t-1} + i^t \cdot \check{c}^t, \quad (21)$$

Выходной фильтр («output gate») аналогичен двум предыдущим и имеет вид:

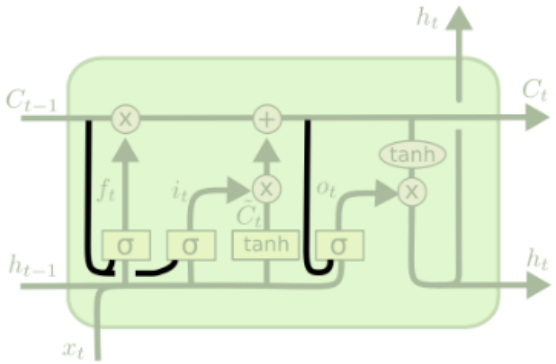
$$o^t = \sigma(W^o x^t + U^o h^{t-1}), \quad (22)$$

Итоговое значение LSTM-слоя определяется выходным фильтром (22) и нелинейной трансформацией над состоянием блока памяти (23):

$$h^t = o^t \cdot \tanh\,(c^t), \quad (23)$$

Существует множество вариантов используемых каждым слоем функций активации, возможны некоторые небольшие изменения самой схемы и каких-либо её параметров. Однако при этом суть функционирования не меняется – сначала фильтруют часть памяти, затем запоминают часть нового сигнала, и уже потом на основе этих данных вычисляется результат. Например, один из популярных вариантов LSTM-сети представлен на рисунке 4. Его

предложили в 2000 году Феликс Герс и Юрген Шмидхубер в своей научной работе.



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

Рисунок 4 – Схематическое изображение LSTM модели Герса и Шмидхубера

2.1.2 GRU

Рассмотренная ранее рекуррентная сеть LSTM вводит достаточно большое количество различных дополнительных параметров в текущую сеть. Таким образом получается, что в существующей сети есть 8 наборов весов, для каждого из 4-х ворот в каждом блоке. В простом рекуррентном модуле, рассмотренном ранее, их было бы всего 2. Обучение этих дополнительных наборов весов влечет за собой более высокую стоимость обучения, поэтому в 2014 году в одной из научных работ [8], была представлена модель GRU (Gated Recurrent Unit), основанная на тех же принципах, что и LSTM, но использующая меньше фильтров и операций для вычисления h^t [9]. Фильтр обновления z^t (update gate) и фильтр сброса состояния r^t (reset gate) вычисляются по следующим формулам:

$$z^t = \sigma(W^z x^t + U^z h^{t-1}), \tag{24}$$

$$r^t = \sigma(W^r x^t + U^r h^{t-1}), \tag{25}$$

Выходное значение h^t вычисляется на основе промежуточного значения $\tilde{h}^t$, которое при помощи фильтра сброса состояния (25) определяет, какие

значения предыдущего шага h^{t-1} следует исключить (здесь можно видеть прямую аналогию с фильтром забывания из LSTM):

$$\tilde{h}^t = \tanh(Wx^t + r^t \cdot Uh^{t-1}), \quad (26)$$

Используя фильтр обновления (24) и промежуточное значение (26), имеем:

$$h^t = z^t \cdot h^{t-1} + (1 - z^t) \cdot \tilde{h}^t \quad (27)$$

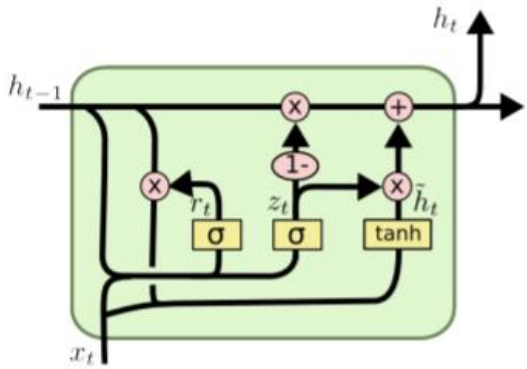


Рисунок 5 – Схематическое изображение ячейки GRU

Стоит отметить, что такое активное внимание к рассмотренному семейству моделей в настоящее время обусловлено, в частности, высокими показателями их эффективности при решении многих задачах. Как и другие рекуррентные нейронные сети, LSTM и GRU (особенно двух- и многослойные) характеризуются достаточно сложной процедурой обучения. Значительно ускорить процессы обучения глубоких нейросетей позволяют графические процессоры (Graphics Processing Unit, GPU), что наглядно демонстрируется активной реализацией (и оптимизацией) описанных рекуррентных моделей под GPU-вычисления.

В заключение можно констатировать, что рекуррентные нейросетевые модели, активно представленные в последнее время в виде LSTM и GRU,

являются эффективными и перспективными алгоритмами машинного обучения для широкого спектра прикладных задач.

2.2 Transformer

Популярным подходом для языкового моделирования является использование рекуррентных нейронных сетей, поскольку они хорошо фиксируют зависимости между словами, особенно при использовании таких модулей, как LSTM. Однако рекуррентные нейронные сети имеют тенденцию быть медленными, и их способность изучать долгосрочные зависимости все еще ограничена из-за исчезающих градиентов [5].

«Transformer» - модель, изобретенная в 2017 году, которая использует механизм «attention» для повышения скорости, с которой можно обучать эти модели. Вместо того, чтобы обрабатывать токены один за другим, «attention»-слои получают сегмент токенов и изучают зависимости между ними одновременно, используя матрицы весов, сформированные в процессе обучения для получения абстрактных векторов: «Query», «Key» и «Value», которые формируют «Attention Head». Сеть трансформера состоит из нескольких слоев, каждый из которых имеет несколько «Attention Head», используемых для изучения различных взаимосвязей между токенами [5, 10, 11].

Как и во многих лингвистических моделях, входные слова сначала преобразуются в векторное представление. Из-за параллельной обработки в «attention»-слое модели также необходимо добавить информацию о порядке следования токенов, шаг с именем «Positional Encoding», который помогает сети узнать их положение. В общем виде, этот шаг выполняется с помощью синусоидальной функции, которая генерирует вектор в соответствии с положением токена без каких-либо изученных параметров.

Трансформеры могут принимать на вход последовательность любых элементов: слова, предложения, изображения и так далее, результатом работы

трансформера так же является последовательность элементов, поэтому далее будем рассматривать модель трансформера, которая принимает на вход последовательность R элементов и возвращать последовательность T элементов. В данном примере будем считать, что на вход подается последовательность слов в виде предложения.

2.2.1 Высокоуровневое представление архитектуры

Трансформер состоит из двух основных частей: кодирующей («encoder») и декодирующей («decoder»). Кодирующий элемент обрабатывает каждый элемент во входной последовательности и накапливает информацию, которую он захватывает в матрицу, называемой контекстом. После обработки всей входной последовательности кодирующая часть формирует контекст и отправляет его декодирующей части, которая начинает последовательно создавать выходную последовательность. Высокоуровневое представление модели трансформер представлено на рисунке 6.

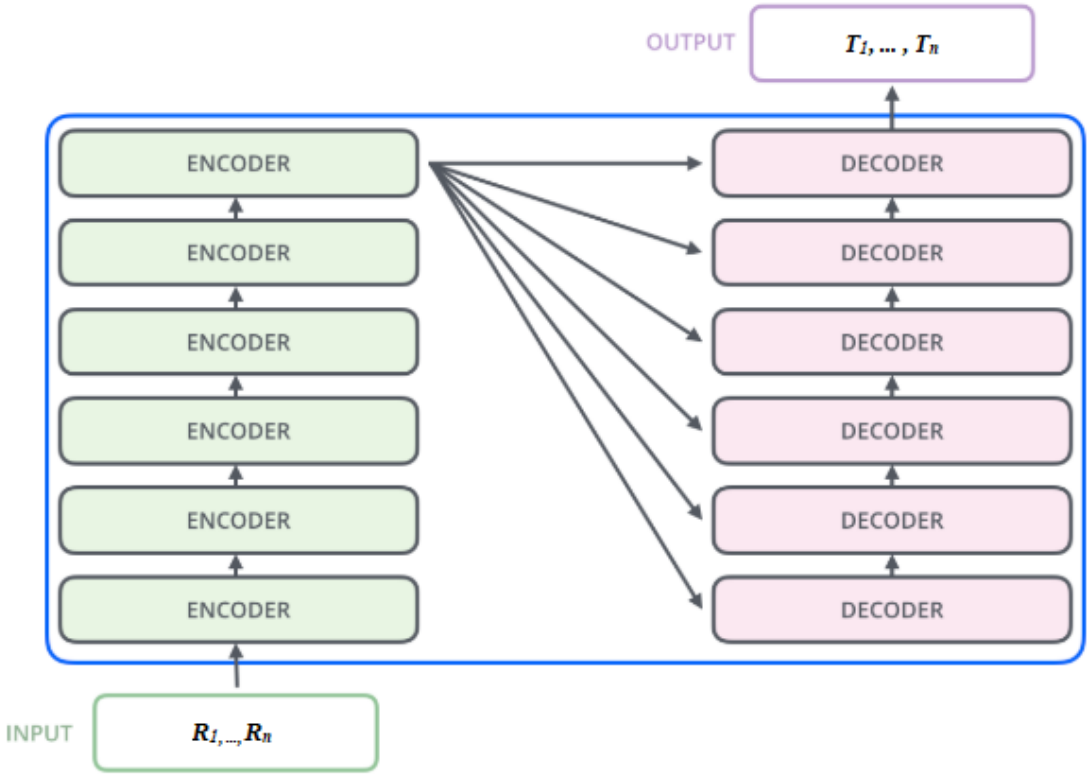


Рисунок 6 – Высокоуровневое представление модели трансформер

На рисунке 6 видно, что кодирующий и декодирующий элемент на самом деле представляют из себя множество элементов с идентичной структурой, структура кодирующего и декодирующего элемента различается, но при этом каждый экземпляр имеет свои собственные веса. Каждый кодирующей элемент можно разделить на два слоя, а декодирующей на три.

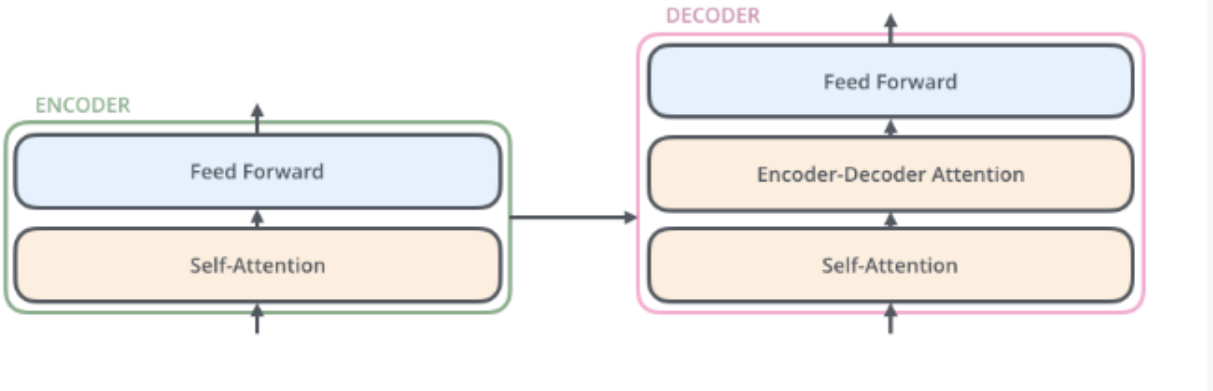


Рисунок 7 – Высокоуровневое представление кодирующего и декодирующего элемента модели «Transformer»

Сначала входная последовательность *R* поступает на «self-attention» слой кодирующего элемента, который позволяет при кодировании конкретного элемента последовательности *R* обращать также внимание и на остальные элементы последовательности. Результат, полученный из слоя «self-attention», отправляется дальше в нейронную сеть прямого распространения «feed forward», которая в итоге сформирует матрицу контекста. Декодирующий элемент содержит перечисленные ранее слои, но между ними находится вспомогательны слой, который помогает фокусироваться на релевантных частях последовательности *R*.

2.2.2 Кодирующий элемент

В данном подразделе будет рассмотрена работа кодирующего элемента: какие процессы происходят внутри и как последовательность *R* преобразуется в матрицу контекста.

Сначала каждое слово переносится в векторное пространство, преобразуюсь в массив признаков размерностью 512. Размерность вектора признаков является гиперпараметром, который можно задать, но в основном он равняется длине самого длинного предложения в обучающей выборке модели. После того как преобразовали слова, каждое из них проходит «self-attention»- и «feed forward»-слои. Каждое слово идет по своему собственному пути, но в слое «self-attention» у нас присутствует зависимость этих путей, а именно их порядок в последовательности. В пункте 2.2.3 будет рассмотрена более подробно последовательность слов в «self-attention», однако в слое «feed forward» нет этих зависимостей, что позволяет распараллелить данный процесс, благодаря чему и достигается высокая скорость трансформера.

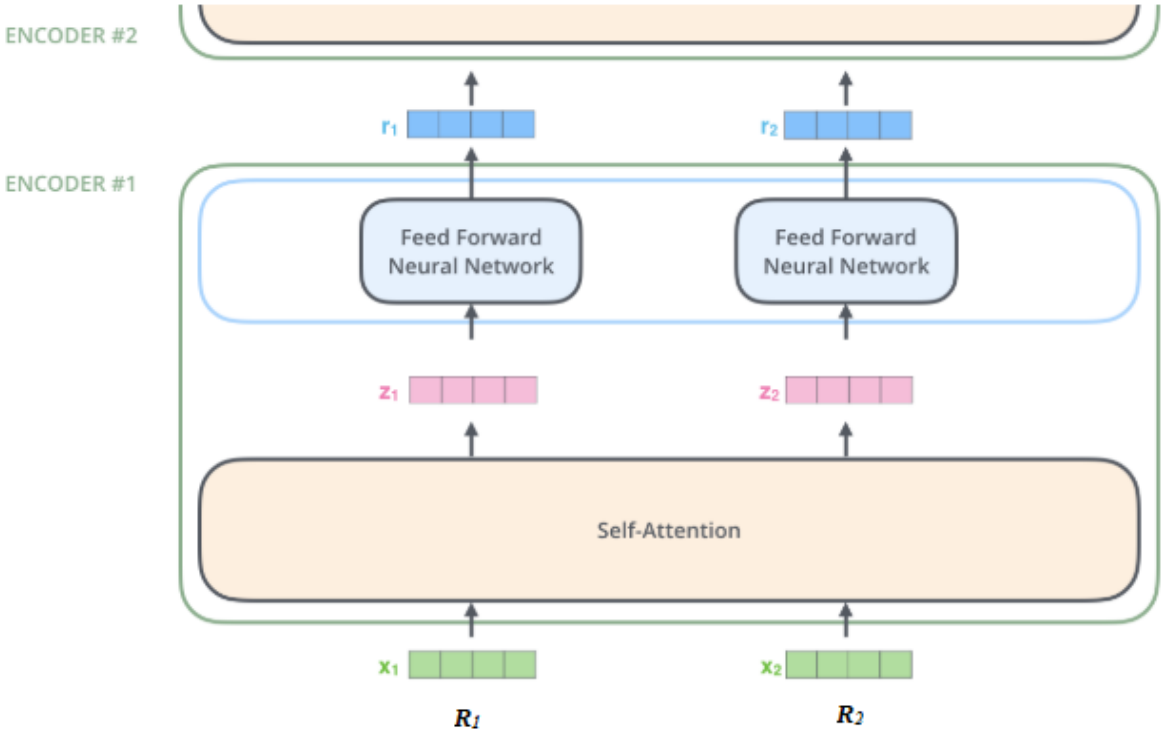


Рисунок 8 – Процесс распараллеливания в кодирующем элементе

При обработке входной последовательности слой «self-attention» помогает модели захватить контекст, оглядываясь на другие позиции входной последовательности, и найти дополнительную информацию, которая поможет лучше закодировать данное слово.

Рекуррентные нейронные сети также позволяют захватывать информацию не только из кодирующего элемента, но и из других элементов входной последовательность, но они позволяет захватить контекст только относительно одной стороны последовательности: справа или слева. Трансформеры же благодаря слою «self-attention» позволяют захватить контекст из всей входной последовательности как слева, так и справа одновременно, что является очень важным для лингвистических моделей, так как позволит более качественно распределять вероятностью для предсказываемого слова [5, 11]. Далее рассмотрим работу слоя «self-attention» более внимательно, поэтапно.

Первый этап – это получение трех векторов из каждого элемента входной последовательности: вектор запроса «Query», вектор ключа «Key» и вектор значений «Value». Перечисленные вектора получается в процессе перемножения матрицы векторного представления слова «Embedding» на матрицы полученные во время процесса обучения: W^Q, W^K, W^V.

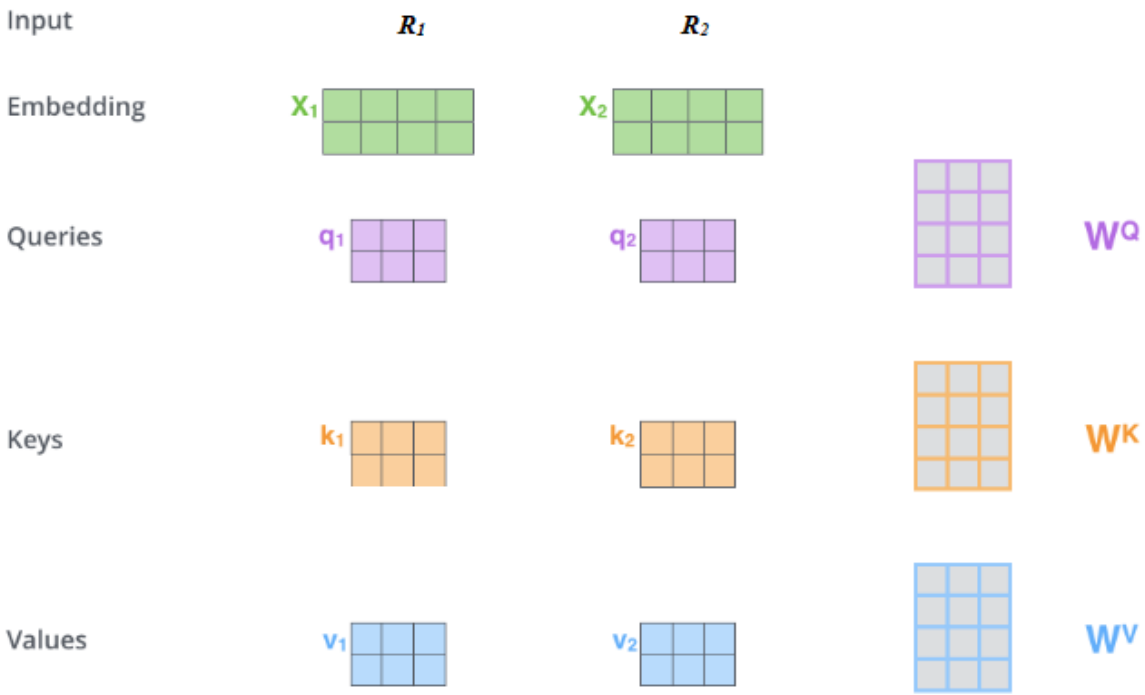


Рисунок 9 – Первый этап слоя self-attention

Второй этап – это расчёт коэффициента «score». Необходимо оценить каждое слово во входной последовательности относительно слова, для которого вычисляем значение на выходе из слоя «self-attention». Коэффициент позволяет определить, как сильно влияют значения других слов на слово в конкретной позиции. Коэффициент рассчитывается с помощью перемножения вектора запроса слова в конкретной позиции и вектора ключа слова, влияние которого нужно узнать. Если рассчитываем значения коэффициента «score» для элемента R_1, то первый коэффициент будет равняться скалярному произведению Q_1 и K_1, второй Q_1 и K_1 , и так далее до Q_1 и K_n. На рисунке 10 проиллюстрирован процесс расчет коэффициента для элемента R_1, для значений коэффициентов используются случайные значения.

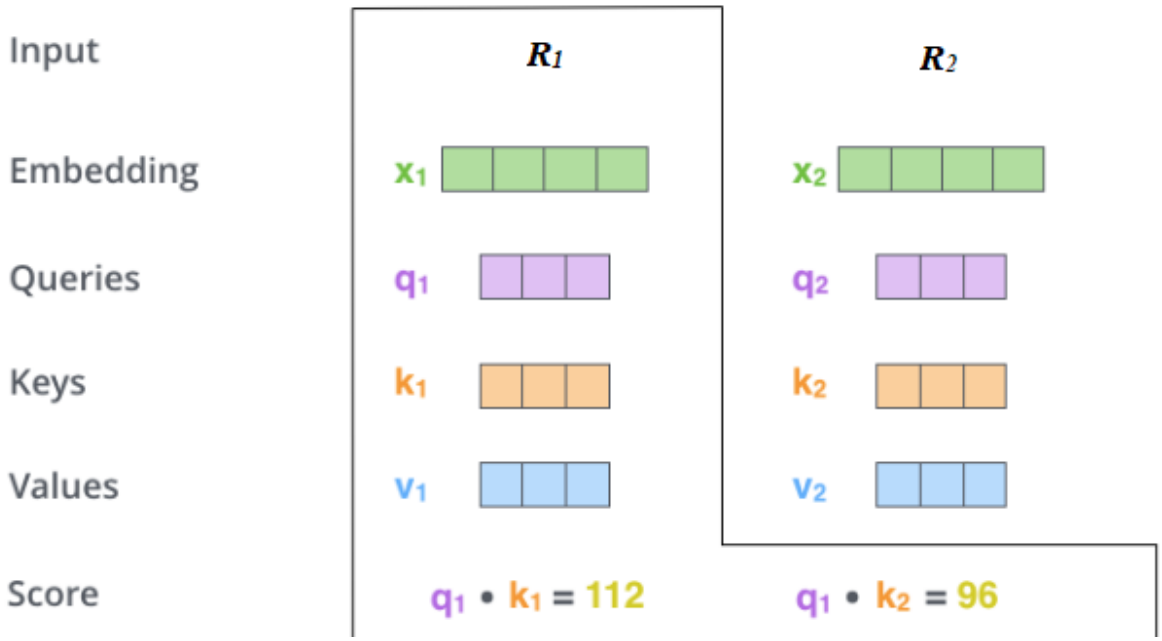


Рисунок 10 – Расчёт коэффициента Score для слова R_1

Третий и четвертый этап – разделение коэффициентов, полученные на прошлом этапе, на число 8, которое является квадратным корнем размерности вектора «Key» (по умолчанию используется значение 64). После деления на 8 необходимо нормализовать полученные результаты с помощью формулы (7) «softmax». Полученные значения после нормализации определяют влияние каждого слова на слово, для которого рассчитываем значение на выходе слоя «self-attention».

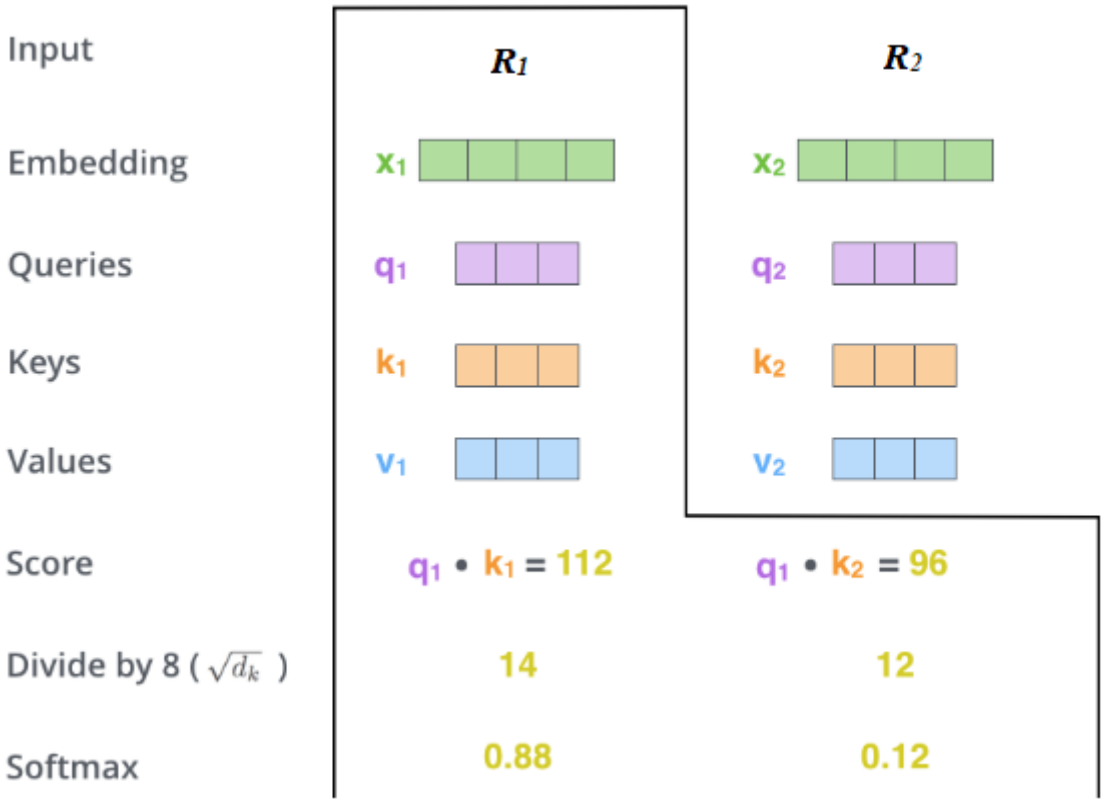


Рисунок 11 – Третий и четвертый этап слоя self-attention

Пятый и шестой этап – необходимость умножения каждого вектора значения на соответствующий нормализованный коэффициент. Необходимо держать без изменений значение слова, для которого идет расчет значения слоя «self-attention», и при этом отвести на второй план слова, которые оказывают наименьшее влияние на это слово. Результаты перемножения всех векторов значения и нормализованных коэффициентов необходимо сложить, полученное значения будет являться значением для данного слова на выходе из слоя «self-attention».

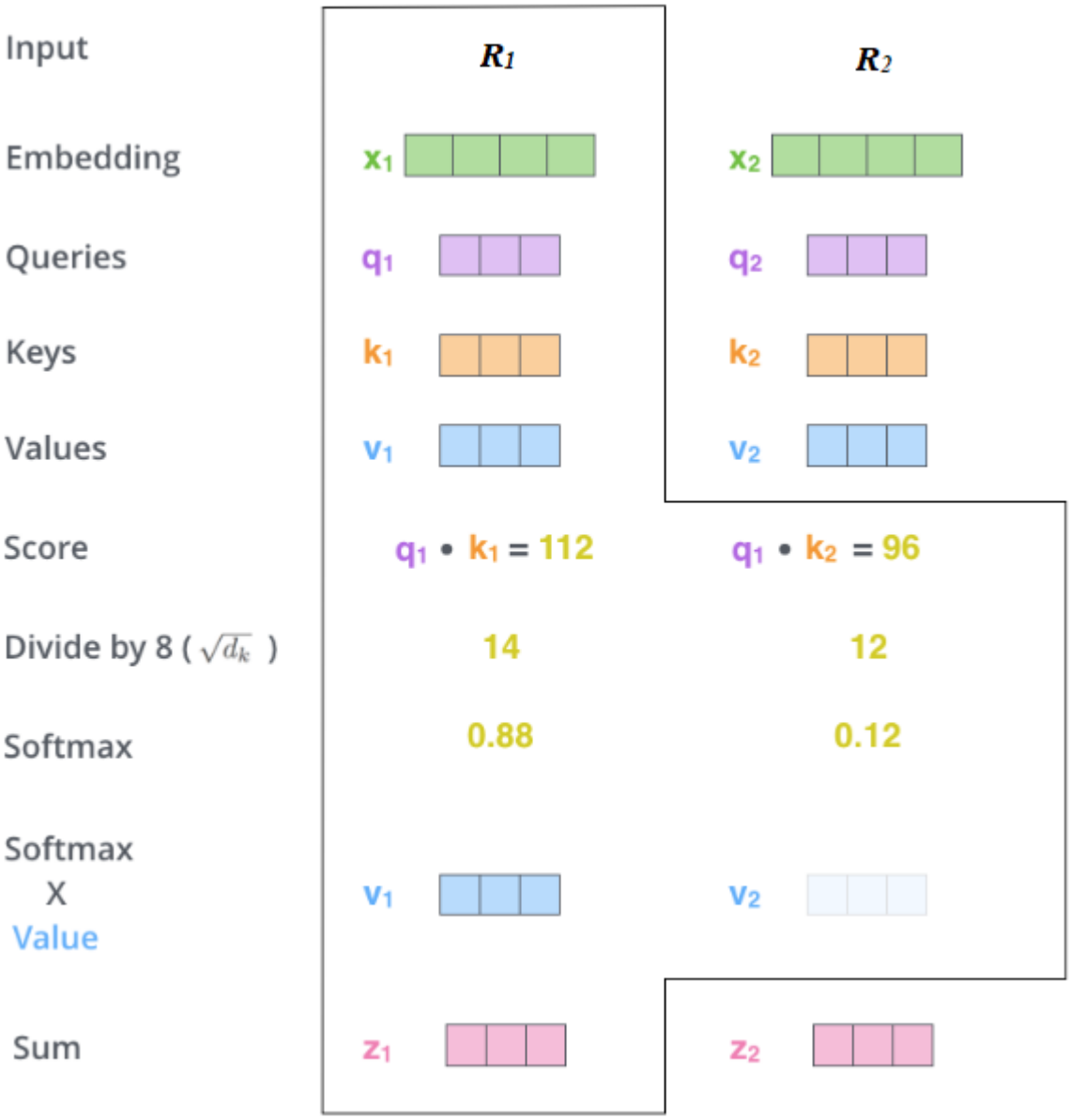


Рисунок 12 – Пятый и шестой этап слоя self-attention

В результате будет получен вектор, который затем отправится в «feed forward»-слой. Для более быстрой обработки всей последовательности входных элементов в реализации используются матричные формы представленных выше векторов. На рисунке 13 проиллюстрирован процесс получения значения слоя «self-attention» со второго по шестой этап в матричной форме.

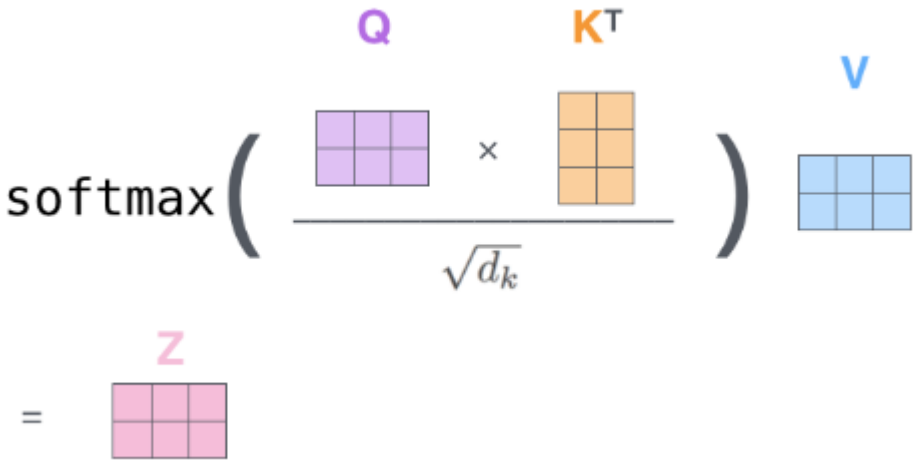


Рисунок 13 – Матричная форма с 2 по 6 этапа слоя self-attention

2.2.3 Порядок во входящей последовательности

Для слоя «self-attention» очень важен порядок слов для понимания контекста (если, допустим, слово «как» стоит в начале предложения, то скорее всего это будет вопросительное предложение и наоборот, если слово «как» стоит в начале или где-нибудь в конце, то это маловероятно). Поэтому в модели «Transformer» к векторному представлению слова добавляется позиционный вектор той же размерности, который создает осмысленное расстояние между векторными представлениями в процессе их проецирования в векторы *Q*, *K*, *V* и конечному значению на слое «self-attention».

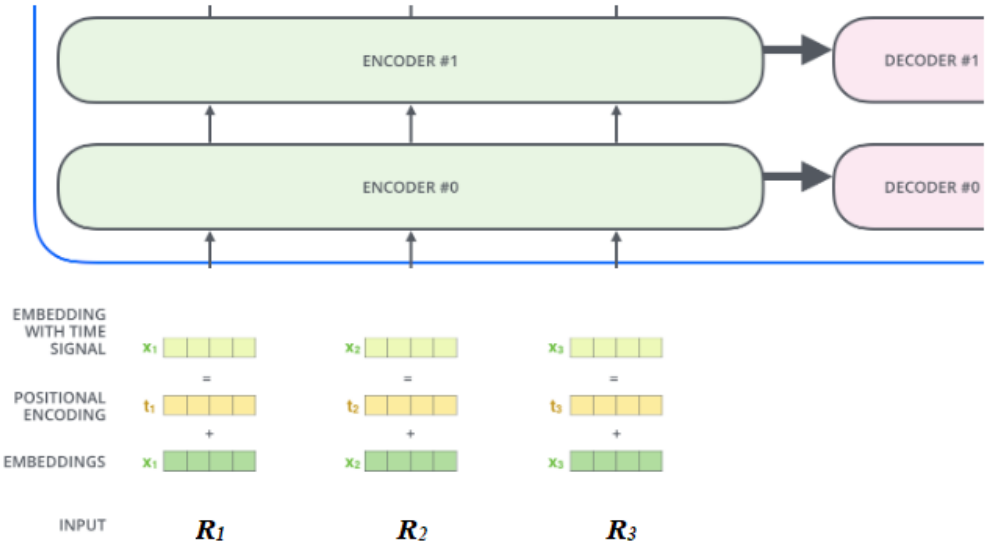


Рисунок 14 – Добавление позиционного вектора

На рисунке 15 представлены значения позиционных векторов в виде тепловой карты для 20 элементов, где каждая строка соответствует позиционному вектору: первую строку добавляем к первому элементу последовательности, вторую ко второму элементу, и так далее. Каждая строка содержит 512 значений от -1 до 1.

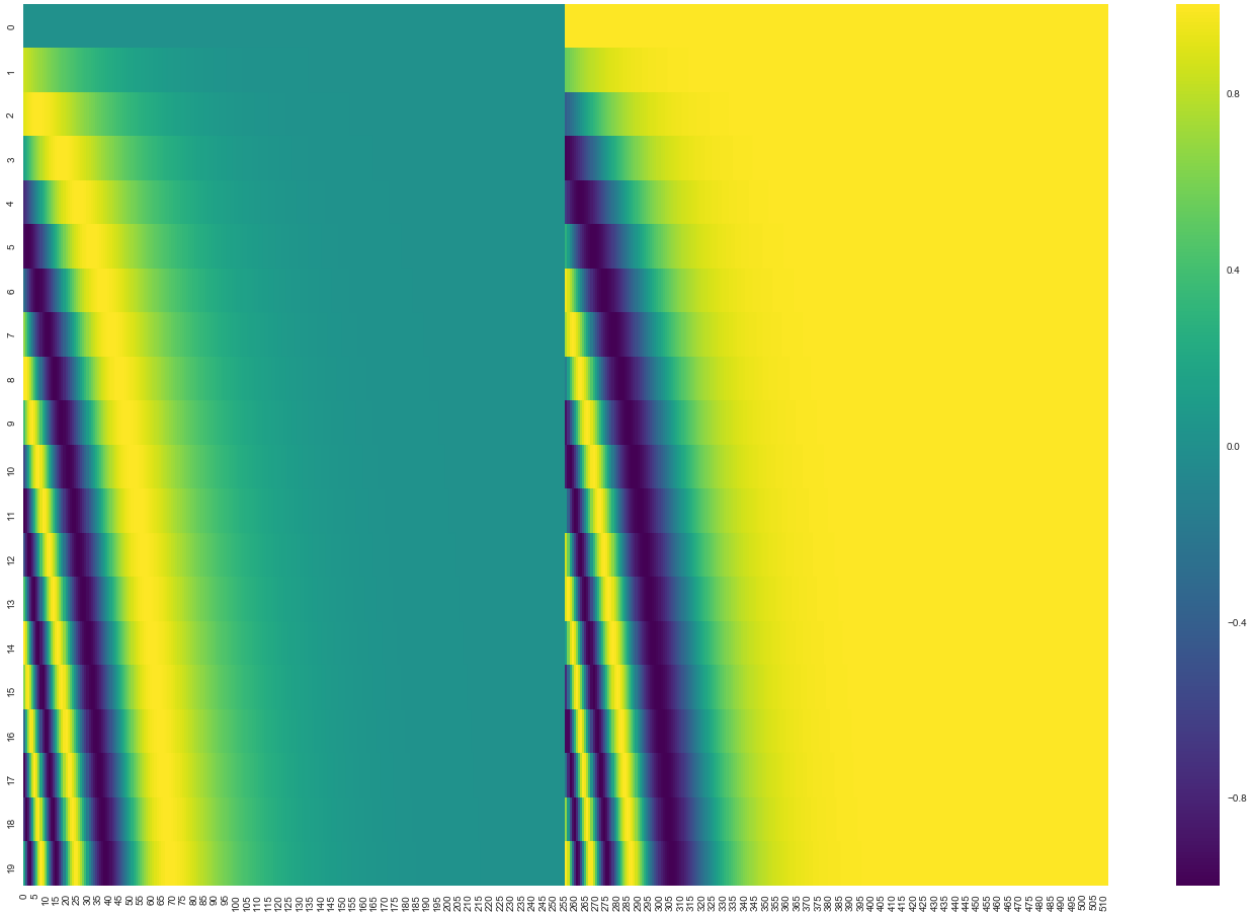


Рисунок 15 – Тепловая карта значений позиционных векторов

2.2.4 Остаточная связь

Последней не разобранной деталью кодирующего элемента является остаточная связь. Каждый кодирующий элемент имеет вокруг себя остаточную связь, после которой наступает этап нормализации слоя. Для каждого слова необходимо сложить вектор, сформированный после добавления позиционного вектора, с вектором, полученным из слоя «self-attention», и после этого провести этап нормализации слоя. Так же остаточная связь присутствует и в

декодирующем элементе. Полная схема работы кодирующего и декодирующего элемента представлена на рисунке 16.

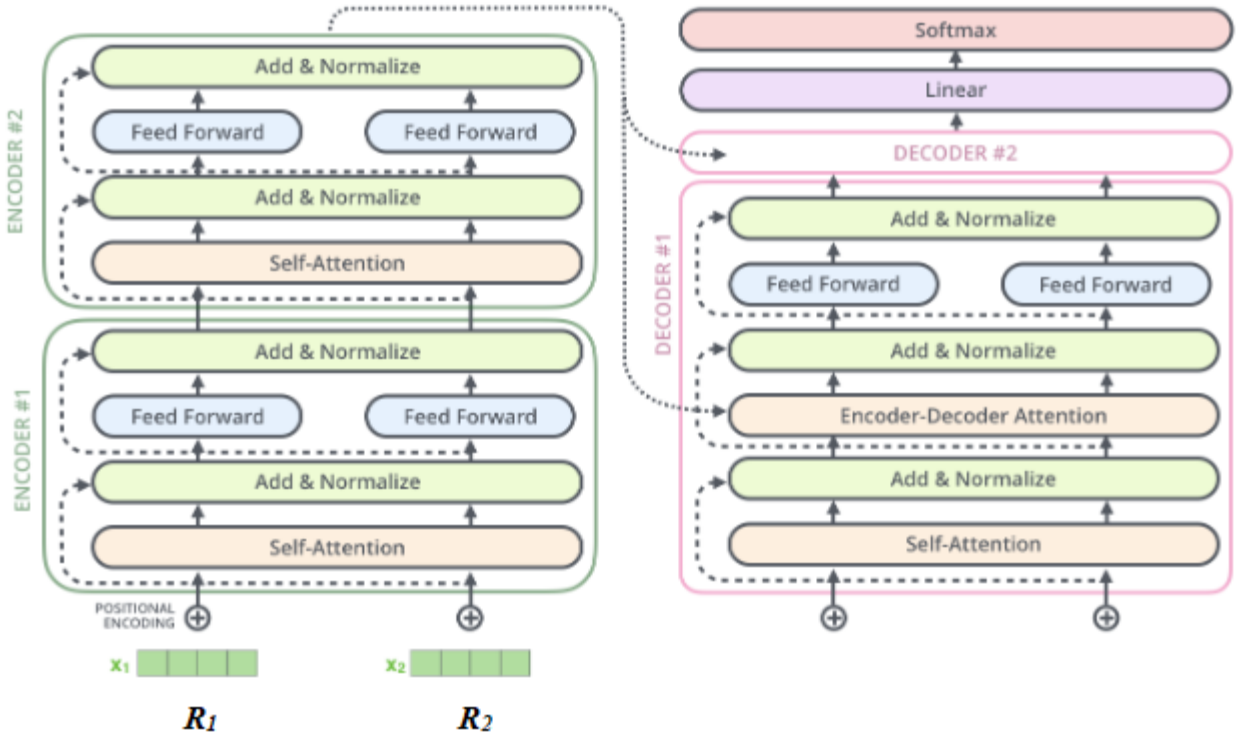


Рисунок 16 – Схема работы кодирующего и декодирующего элемента

2.2.5 Декодирующий элемент

Кодирующий элемент начинает обрабатывать входную последовательность, далее результат работы последнего кодирующего элемента преобразуется в набор векторов *K* и *V*, которые передаются в каждый декодирующий элемент. Полученные векторы из кодирующего элемента используются всеми декодирующими элемента в их «encoder-decoder attention»-слое, помогая в понимании контекста. Начальное значение вектора *Q*, которое используется для вычисления значения на выходе каждого декодирующего элемента, для первого элемента выходной последовательности определяется в процессе обучения, для каждого последующего элемента значение вычисляется на основе предыдущего значения вектора *Q* и значения, полученного на выходе декодирующего элемента на предыдущем этапе. Каждый этап декодирующего элемента возвращает элемент выходной последовательности, следующий этап

начинается с формирования контекста с помощью слоя «self-attention» для слов, полученных на предыдущем этапе.

2.2.6 Заключительный этап трансформера

Декодирующий элемент на выходе возвращает вектор чисел с плавающей точкой. Для того чтобы преобразовать этот вектор в конечную форму элемента выходной последовательности *T*, используется линейный слой и последующая нормализация с помощью функции (7) «softmax». Линейный слой представляет собой простую полносвязную нейронную сеть, которая переводит вектор, созданный декодирующим элементом, в более большой вектор, называемый логит-вектором. Логит-вектор для лингвистических моделей имеет размерность, равную количеству слов в нашей модели. Таким образом, интерпретируется результат работы декодирующего элемента [10]. Последующая нормализация позволяет получить распределение вероятностей для каждого слова в словаре модели, и на выход лингвистической модели типа трансформер выбирается слово с наибольшей вероятностью.

2.3 BERT

BERT (Bidirectional Encoder Representations from Transformers) – это модель, побившая несколько рекордов по качеству решения ряда задач обработки естественного языка. Вскоре после выхода статьи [12], описывающей модель, команда разработчиков также выложила в открытый доступ код модели и сделала возможным скачивание различных версий модели BERT [13], которые уже имели предварительное обучение на больших наборах данных. Этот шаг позволяет любому разработчику встраивать в свои модели машинного обучения для обработки естественного языка уже готовый функциональный компонент, сохраняя время и ресурсы, необходимые для обучения модели обработки языка с нуля [12, 13].

Релизная статья [12] описывает две модели BERT разных размеров:

- «BERT BASE» (базовая);
- «BERT LARGE» (расширенная).

BERT представляет из себя набор кодирующих элементов трансформера.

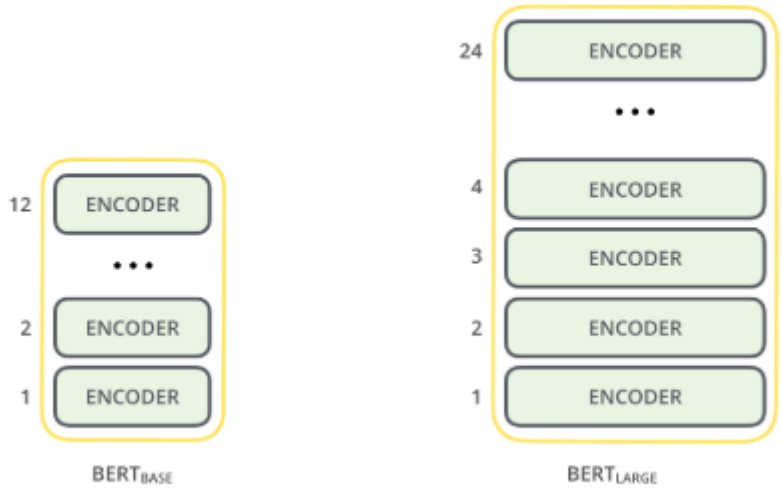


Рисунок 17 – Архитектура BERT

Обе модели BERT содержат большое количество кодирующих элементов: 12 для базовой версии и 24 для расширенной. Они также имеют более крупные слои сети прямого распространения: 768 и 1024 скрытых нейронов соответственно. Больше слоев внимания «attention heads» (12 и 16 соответственно), чем в базовой конфигурации трансформера, описанной в исходной статье [12]: 6 кодирующих элементов, 512 скрытых нейронов, 8 слоев внимания.

Точно так же, как и в обычном трансформере, BERT принимает на вход последовательность слов, которая затем продвигается вверх через кодирующие элементы. Каждый кодирующий элемент применяет слой «self-attention» и передает результаты в сеть прямого распространения, после чего направляет его следующему кодирующему элементу. Для каждой позиции на выход

подается вектор размерностью равной количеству скрытых нейронов в сети прямого распространения, 768 в базовой модели BERT [13].

Для того чтобы получить распределение вероятностей для конкретного слова, необходимо замаскировать это слово с помощью маски «{MASK}», и получить распределение вероятностей для этой маски. Использование масок нужно для того, чтобы исключить влияние самого слова на контекст, при этом замаскировать можно не больше 15% от всей входной последовательности.

3 Реализация алгоритма

Реализация алгоритма будет осуществляться на нейросетевой модели BERT. Поскольку обучение данной модели является очень длительным и дорогостоящим процессом, будет использована модель с предварительным обучением. Существует множество различных вариаций этой модели, но для русского языка выбор сильно ограничен, поэтому будет использоваться версия «BERT BASE CASED» (базовая регистрозависимая). Также для оценки достигнутых результатов с помощью BERT, все действия будут проводиться аналогичным образом на моделях «skipgram» и «fasttext», обученных на корпусе «Тайга». Данные модели, в отличии от BERT, не могут так качественно распознавать контекст.

В некоторых случаях BERT может давать высокую оценку для нерелевантных слов, так как для того, чтобы работать со словами, не представленными в словаре, BERT использует метод на основе «WordPiece»-токенизации [15]. В таком случае, слово, отсутствующее в словаре, постепенно разбивается на более мелкие части. Поэтому необходимо использовать словари, которые будут ограничивать BERT и гарантировать использование слов из словаря для синонимизации текста.

Работу алгоритма можно разделить на несколько этапов:

- получение распределения вероятностей для каждого слова в предложении;
- сопоставление слов из словарей с выдачей BERT;
- определение конечного набора слов, которые могут использоваться в качестве синонимов;
- нормализация набора слов с помощью функции (7) и выбор слова для использования в синонимичном тексте.

Работа алгоритма будет рассмотрена на примере предложения «Он подарил мне блестящий меч». Данное предложение будет подано на вход

нашему алгоритму, при этом предполагается, что это должно хорошо раскрыть возможности модели BERT по работе с контекстно-зависимыми словами.

3.1 Получение распределения вероятностей

Код для получения распределения вероятностей для каждого слова предложения и дальнейшего сохранения результатов в Excel-файл представлен в приложении А. Распределение вероятностей для предложения, выбранного в качестве примера, представлено в таблице Б.1. Из полученных результатов можно сделать вывод, что на данном этапе BERT предоставляет более релевантную выборку.

3.2 Сопоставление слов и определение конечного набора слов

Для того чтобы гарантировать использование слов из словаря, необходимо пройтись по всей выборке, полученной от лингвистической модели, и проверить, имеются ли данные слова в нашем словаре, далее в случае совпадения сохранить это слово и его оценку.

Из таблицы Б.1 видно, что BERT предоставляет слова не в начальной форме, что позволяет ставить в верную форму слова из словаря, которые представлены в начальной форме. Для этого необходимо искать совпадения из словаря и выдаче нейросетевой модели, при этом сравнивать слова необходимо предварительно приведя в начальную форму.

Для данного этапа и для более удобной работы на следующих этапов были написаны следующие классы: «Dictionary», «PredictionsForWord», «PredictionsForText». Код этих классов и их методов представлен в приложении В. В таблице 2 представлен результат работы данного этапа в виде конечного набора слов, который будет использоваться для синонимизации входного текста.

Таблица 2 – Результат этапа сопоставления

	«Он»		«мне»		«блестящий»	
	Предположение	Оценка	Предположение	Оценка	Предположение	Оценка
BERT	–	–	новый	4,50	новый	7,04
	–	–	серебрянный	4,72	железный	4,80
	–	–	золотой	3,91	серебрянный	7,25
	–	–	хороший	4,44	золотой	7,30
	–	–	прекрасный	4,22	хороший	5,45
	–	–	–	–	волшебный	7,78
	–	–	–	–	отличный	5,31
fasttext	новый	0,22	–	–	серебрянный	7,08

Из таблицы 2 можно сделать вывод, что модель «skipgram» не справилась с задачей и не сможет синонимизировать данное предложение, а модель «fasttext» предлагает по одному варианту для двух слов, что нельзя назвать хорошим результатом, так как важно создание множества синонимичных текстов.

По таблице 2 видно, что BERT предлагает использовать слова из разных словарей:

- словарь с положительными оценками: «хороший», «лучший», и т.д.;
- словарь с материалами: «деревянный», «железный», и т.д.

Необходимо выбрать один словарь, слова из которого попадут в конечную выборку для синонимизации. Для этого подсчитаем, из какого словаря нейросетевая модель предлагают больше слов. В данном случае таким словарем является словарь с положительными оценками.

3.3 Нормализация и конечный результат

После этапа сопоставления слов имеются следующие выборки: для слова «мне» [«золотой», «хороший», «прекрасный»], для слова «блестящий»

[«золотой», «хороший», «волшебный», «прекрасный», «отличный»]. После процесса нормализации, используя функцию (7) оценок каждого вектора, получается следующие распределение вероятностей [0.33527086, 0.20827057, 0.45645856] и [0.30153937, 0.04756725, 0.49040125, 0.11933872, 0.04115341]. Далее полученные вектора передаются в функцию «numpy.random.choice», которая возвращает одно слово случайным образом, учитывая распределение вероятностей переданных в нее слов, что позволит в дальнейшем алгоритму чаще использовать для синонимизации слова с большей вероятностью.

В результате работы алгоритма для предложения «Он подарил мне блестящий меч» были сгенерированы несколько новых синонимичных предложений. Алгоритм с использованием нейросетевой модели BERT создал два следующих предложения: «Он подарил хороший волшебный меч», «Он подарил золотой волшебный меч». Алгоритм с использованием моделей «skipgram» не смог сгенерировать новое синонимичное предложение, а с моделью «fasttext» получилось следующие предложение: «новый подарил мне серебрянный меч».

4 Финансовый менеджмент, ресурсоэффективность и ресурсосбережение

Выполнение грамотной научно-исследовательской работы требует наличия экономической оценки всех её элементов: как объекта исследования, так и методов, которые для этого используются. Таким образом, целью данного раздела является комплексное описание и анализ финансово-экономических аспектов алгоритма для генерации синонимичных текстов на основе нейросетевой модели. Для достижения поставленной цели необходимо выполнить следующие задачи:

- провести SWOT-анализ;
- определить эффективность исследования
- провести планирование научно-исследовательской работы;
- произвести расчёт бюджета научно-исследовательской работы;
- составить оценку научно-технического эффекта.

4.1 Организация и планирование работы

При организации процесса реализации данного исследования необходимо рационально планировать занятость каждого из его участников и сроки проведения отдельных работ.

В данном пункте составляется полный перечень проводимых работ, определяются их исполнители и рациональная продолжительность. Так как число исполнителей редко превышает двух в большинстве случаев, то для наглядного результата чаще пользуются линейным графиком. Для построения такого графика приведем в таблице 3 перечень работ и занятость исполнителей.

Таблица 3 – Перечень работ и продолжительность их выполнения

№ Этапа	Этапы работы	Исполнители	Загрузка исполнителей
1	Постановка целей и задач, получение исходных данных	Научный руководитель	НР – 100%
2	Составление и утверждение ТЗ	Научный руководитель, студент	НР – 100% С – 10%
3	Подбор и изучение материалов по тематике	Научный руководитель, студент	НР – 50% С – 100%
4	Разработка календарного плана	Научный руководитель, студент	НР – 100% С – 10%
5	Обсуждение литературы	Научный руководитель, студент	НР – 30% С – 100%
6	Написание программы	Студент	С – 100%
7	Тестирование программы	Студент	С – 100%
8	Оформление расчетно-пояснительной записки	Студент	С – 100%
9	Оформление графического материала	Студент	С – 100%
10	Анализ полученных результатов	Научный руководитель, студент	НР – 60% С – 100%

4.1.1 Продолжительность этапов работ

Трудовые затраты в большинстве случаев образуют основную часть стоимости разработки, поэтому важным моментом является определение трудоемкости работ каждого из участников научного исследования.

Трудоемкость выполнения проекта оценивается экспертным путем в человеко-днях и носит вероятностный характер, т.к. зависит от множества

трудно учитываемых факторов. Для определения ожидаемого (среднего) значения трудоемкости $t_{ож}$ используется следующая формула:

$$t_{ожс} = \frac{3 \cdot t_{min} + 2 \cdot t_{max}}{5}, \quad (28)$$

где $t_{ож}$– ожидаемая трудоемкость выполнения i-ой работы чел.-дн.;

t_{min} – минимальная продолжительность работы, дн.;

t_{max}– максимальная продолжительность работы, дн.

Для выполнения перечисленных в таблице 3 работ, требуется группа специалистов из следующего состава:

– Студент (С), соискатель степени бакалавра;

– Научный руководитель (НР).

Исходя из ожидаемой трудоемкости работ, определяется продолжительность каждой работы в рабочих днях $T_{рд}$, учитывающая параллельность выполнения работ несколькими исполнителями. Так, для построения линейного графика необходимо рассчитать длительность этапов в рабочих днях, а затем перевести ее в календарные дни. Расчет продолжительности выполнения каждого этапа в рабочих днях ($T_{рд}$ ведется по формуле:

$$T_{рд} = \frac{t_{ож}}{К_{ВН}} \cdot K_{Д}, \quad (29)$$

где $t_{ож}$ – продолжительность работы, дн.;

$К_{ВН}$– коэффициент выполнения работ, учитывающий влияние внешних факторов на соблюдение предварительно определенных длительностей, в частности, возможно $K_{вн}$ =1;

$K_{Д}$– коэффициент, учитывающий дополнительное время на компенсацию непредвиденных задержек и согласование работ ($K_{Д}$ = 1–1,2; в этих границах конкретное значение принимает сам исполнитель). Возьмем значение $К_{Д} = 1$.

Продолжительность этапов работ и их трудоемкости по исполнителям, занятым на каждом этапе представлена в таблице 4.

Таблица 4 – Временные показатели проведения научного исследования

№ Этапа	Исполнители	Продолжительность работ, дни			Трудоемкость работ, дни			
					$T_{рд}$		$T_{КД}$	
		t_{min}	t_{max}	$t_{ож}$	НР	С	НР	С
1	Научный руководитель	1	2	1,6	1,6	-	1,92	-
2	Научный руководитель, студент	5	10	7	7	0.7	8,4	0,84
3	Научный руководитель, студент	10	15	12	6	12	7,2	14,4
4	Научный руководитель, студент	5	10	7	7	0,7	8,4	0,84
5	Научный руководитель, студент	1	2	1,6	1,6	0,48	1,92	0,58
6	Студент	15	20	17	-	17	-	20,4
7	Студент	3	5	3,8	-	3,8	-	4,56
8	Студент	10	20	14	-	14	-	16,8
9	Студент	1	2	1,6	-	1,6	-	1,92
10	Научный руководитель, студент	5	10	7	4,2	7	5,04	8,4
Итого:				**72,6**	**27,4**	**57,28**	**32,88**	**68,74**

4.1.2 Разработка графика проведения научного исследования

Выполнение ВКР является небольшим по объему исследованием, поэтому наиболее удобным и наглядным является построение ленточного графика проведения научных работ в форме диаграммы Ганта.

Так, построим ленточный график. Для удобства построения графика, длительность каждого из этапов работ из рабочих дней следует перевести в календарные дни. Для этого необходимо воспользоваться следующей формулой:

$$T_{КД} = T_{РД} \cdot T_{К}, \quad (30)$$

где $T_{КД}$ – продолжительность выполнения этапа в календарных днях;

$T_К$ – коэффициент календарности, позволяющий перейти от длительности работ в рабочих днях к их аналогам в календарных днях, который определяется по следующей формуле:

$$\mathrm{T_К} = \frac{\mathrm{T_{КАЛ}}}{\mathrm{T_{КАЛ}} - \mathrm{T_{ВД}} - \mathrm{T_{ПД}}}, \quad (31)$$

где $T_{КАЛ}$ – календарные дни ($T_{КАЛ}$ = 365);

$T_{ВД}$ – выходные дни ($T_{ВД}$ = 52 для при шестидневной рабочей неделе);

$T_{ПД}$ – праздничные дни ($T_{ПД}$ = 10).

$$\mathrm{T_К} = \frac{365}{365 - 52 - 10} = 1{,}20, \quad (32)$$

Таким образом, коэффициент календарности $\mathrm{T_К}$ равен 1,20.

Величины трудоемкости этапов по исполнителям $\mathrm{T_{КД}}$ (данные столбцов 8 и 9 кроме итогов) позволяют построить линейный график осуществления проекта.

Пример построения линейного графика приведен в таблице 5.

Таблица 5 – Линейный график работ

Этап	Вид работ	НР	С	Продолжительность выполнения работ								
				март			апрель			май		
				10	20	30	10	20	30	10	20	30
1	Постановка целей и задач, получение исходных данных	1,92	-									
2	Составление и утверждение ТЗ	8,4	0,84									
3	Подбор и изучение материалов по тематике	7,2	14,4									
4	Разработка календарного плана	8,4	0,84									
5	Обсуждение литературы	1,92	0,58									
6	Написание программы	-	20,4									

Продолжение таблицы 5

Этап	Вид работ	НР	С	Продолжительность выполнения работ								
				март			апрель			май		
				10	20	30	10	20	30	10	20	30
7	Тестирование программы	-	4,56									
8	Оформление расчетно-пояснительной записки	-	16,8									
9	Оформление графического материала	-	1,92									
10	Анализ полученных результатов	5,04	8,4									

– Научный руководитель;

– Студент.

4.2 SWOT-анализ

SWOT-анализ представляет собой сводную таблицу, иллюстрирующую связь между внутренними и внешними факторами компании. Целью SWOT-анализа является предоставление возможности оценки риска и конкурентоспособности компании или товара в данной отрасли производства.

Методика SWOT-анализа необходима для того, чтобы определить наиболее прозрачное на положение компании, продукции или услуги в данной отрасли.

Приведем матрицу SWOT-анализа для алгоритма генерации синонимичного текста.

Таблица 6 – Матрица SWOT – анализа

	Сильные стороны	**Слабые стороны**
	С1. Высокая степень автоматизации создания синонимичных текстов. С2. Высокая степень понимания контекста алгоритмом. С3. Использование нейросетевой модели.	Сл1. Вероятность генерации слов, нарушающих семантическую целостность текста. Сл2. Необходимость задавать список слов, которые должны присутствовать в сгенерированном тексте.
Возможности **В1.** Автоматизация создания уникальных текстов.	**В1С1С2** Автоматизированный процесс создания уникальных текстов, позволяющий снизить временные и финансовые затраты.	**В1Сл1** Возможна генерация некорректных текстов, вследствие чего требуется ручная проверка результатов. **В1Сл2** Требуется составление списка ключевых слов.
В2. Адаптация алгоритма под иностранные языки.	**В2С3** Использование нейросетевой модели, позволяет легко адаптировать алгоритм для работы не только с русским языком.	**В2Сл1** Методы для сохранения семантической целостности текста зависят от используемого языка.
Угрозы **У1.** Появление более совершенных алгоритмов. **У2.** Плохое качество генерируемых текстов. **У3.** Наличие конкурентов.	**У1С1** Возможность замены, усовершенствование нейросетевой модели. **У3С1С2С3** Обеспечение конкуренции с помощью маркетинга.	**У1Сл1** Способы сохранения семантической целостности текста придется изменить в зависимости от алгоритма**.** **У3У2Сл1Сл2** Невыгодное положение на рынке из-за качества генерируемых текстов и необходимости ручного составления списка ключевых слов.

Таким образом, можно сделать вывод о том, что наиболее эффективными в сложившейся ситуации представляются следующие стратегии:

– по возможности автоматизировать валидацию сгенерированного алгоритмом текста;

– необходимо обеспечить возможность легкой замены методов для сохранения семантической целостности текста.

4.3 Анализ конкурентных решений

Основной целью программы является генерация синонимичных текстов. Такая цель может достигаться путем использования статистических лингвистических моделей. Чем лучше качество сгенерированного текста, тем выше конкурентоспособность. Автоматизация процесса генерации синонимичных текстов позволит снизить финансовые издержки. Анализ конкурентных решений представлен в таблице 7.

Для оценочной карты были выбраны следующие критерии:

– скорость выполнения работы;

– использование необходимых слов в тексте;

– уникальность полученного текста;

– семантическая целостность текста;

– прибыль компании;

Таблица 7 – Оценочная карта для сравнения конкурентных решений

Критерии оценки	Вес критерия	Баллы		Конкурентоспособность	
		К1 (Ручная синонимизация)	К2 (Использование программы)	К1 (Ручная синонимизация)	К2 (Использование программы)
Экономические критерии оценки эффективности					
1. •скорость выполнения работы	0,2	1	4	0,2	0,8
2. •использование необходимых слов в тексте	0,1	5	4	0,5	0,4

Продолжение таблицы 7

Критерии оценки	Вес критерия	Баллы		Конкурентоспособность	
		К1 (Ручная синонимизация)	К2 (Использование программы)	К1 (Ручная синонимизация)	К2 (Использование программы)
Экономические критерии оценки эффективности					
3. •уникальность полученного текста	0,15	4	3	0,6	0,45
4. •семантическая целостность текста	0,25	5	3	1,25	0,75
5. •прибыль компании	0,3	2	5	0,6	1,5
Итого	1	5	2	3,15	**3,9**

Позиция разработки оценивается по каждому показателю экспертным путем по пятибалльной шкале, где 1 – наиболее слабая позиция, а 5 – наиболее сильная. Анализ конкурентных решений определяется по формуле:

$$К = \sum В_i \cdot Б_i, \quad (33)$$

где $К$ – конкурентоспособность решения или конкурента, $В_i$ – вес показателя (в долях единицы), $Б_i$ – балл i-го показателя.

Таким образом, можно сделать вывод, что использование программы является наиболее предпочтительным методом для синонимизации текста (значение 3,9 является максимальным).

4.4 Потенциальные потребители результатов исследований

В процессе написания выпускной квалификационной работы были определены следующие потенциальные потребители разработанного продукта. К ним можно отнести группу лиц, которые занимаются созданием уникальных текстов для коммерческих целей. В свою очередь, эту группу можно разделить

по следующим признакам: количество создаваемых уникальных текстов, требовательны к качеству получаемого текста, финансовые возможности.

Разработан алгоритм синонимизации текста на основе нейросетевой модели. Однако, прежде чем предложить заинтересованным лицам данный продукт, необходимо оценить их возможности и потребности. Для этого проведем классификацию потенциальных клиентов в таблице 8.

Таблица 8. Группы клиентов в зависимости от их возможностей и потребностей

Группы клиентов	Возможности	Потребности	Необходимость в продукте
Группа 1 Фрилансеры	Имеют низкие финансовые возможности	Не требуется большое количество уникальных текстов, главная цель снизить временные затраты	Отсутствует, или низкая.
Группа 2 Малый бизнес	Имеют достаточно финансовых возможностей	Постоянно нуждаются в небольшой количестве качественных уникальных текстов	Средняя
Группа 3 Средний бизнес	Имеют достаточно финансовых возможностей	Основной потребностью является возможность получать большое количестве уникальных текстов.	Высокая
Группа 4 Крупный бизнес	Имеют достаточно финансовых возможностей	Основной потребностью является уменьшение финансовых издержек.	Очень высокая

4.5 Расчет сметы затрат на выполнение проекта

В состав затрат на создание проекта включается величина всех расходов, необходимых для реализации комплекса работ, составляющих содержание данной разработки. Расчет сметной стоимости ее выполнения производится по следующим статьям затрат:

- Материалы и покупные изделия;
- Заработная плата;
- Социальный налог;
- Расходы на электроэнергию (без освещения);

– Амортизационные отчисления;
– Оплата услуг связи;
– Прочие (накладные расходы) расходы.

4.5.1 Расчет материальных затрат

К данной статье расходов относится стоимость материалов, покупных изделий, расходуемых непосредственно в процессе выполнения работ над объектом исследования.

Покажем отражение стоимости всех материалов, используемых при работе над проектом, включая расходы на их приобретение и, при необходимости, доставку. Расчет затрат на материалы производится по форме, приведенной в таблице 9.

Таблица 9 – Материальные затраты

Наименование	Единица измерения	Количество	Цена за ед, руб.	Сумма, руб
Бумага	Пачка	1	300	300
Канцелярские принадлежности	шт.	5	100	500
Картридж для принтера	шт.	1	3000	3000
Итого:				**3800**

Допустим, что ТЗР составляют 5 % от отпускной цены материалов, тогда расходы на материалы с учетом ТЗР равны:

$$С_{мат} = 3\,800 \cdot 1{,}05 = 3\,990 \text{ руб.}$$

4.5.2 Расчет заработной платы для исполнителей

Данная статья расходов включает заработную плату научного руководителя и студента (в его роли выступает исполнитель проекта), а также премии, входящие в фонд заработной платы.

Расчет основной заработной платы выполняется на основе трудоемкости выполнения каждого этапа и величины месячного оклада исполнителя.

Среднедневная тарифная заработная плата ($ЗП_{дн-т}$) рассчитывается по формуле:

$$ЗП_{дн-т} = \frac{МО}{25{,}083}, \quad (34)$$

Учитывающей, что в году 301 рабочий день и, следовательно, в месяце в среднем 25,083 рабочих дня (при шестидневной рабочей неделе).

Пример расчета затрат на полную заработную плату приведены в таблице 10. Затраты времени по каждому исполнителю в рабочих днях с округлением до целого взяты из таблицы 4. Для учета в ее составе премий, дополнительной зарплаты и районной надбавки используется следующий ряд коэффициентов: $К_{ПР}$ = 1,1; $К_{доп.ЗП}$ = 1,188; $К_р$ = 1,3. Таким образом, для перехода от тарифной (базовой) суммы заработка исполнителя, связанной с участием в проекте, к соответствующему полному заработку (зарплатной части сметы) необходимо первую умножить на интегральный коэффициент $К_и$ = 1,1* 1,188*1,3 = 1,699. Вышеуказанное значение $К_{доп.ЗП}$ применяется при шестидневной рабочей неделе, при пятидневной оно равно 1,113, соответственно в этом случае $К_и$ = 1,62.

Таблица 10 – Затраты на заработную плату

Исполнитель	Оклад, руб./мес.	Среднедневная тарифная ставка руб./раб.день	Затраты времени, раб.дни	Коэффициент	Фонд з/платы, руб.
НР	33 664	1 342,09	28	1,699	**63 845,9**
С	15 470	616,75	58	1,62	**57 949,83**
Итого					**121795,73**

4.5.3 Расчет затрат на социальный налог

Затраты на единый социальный налог (ЕСН), включающий в себя отчисления в пенсионный фонд, на социальное и медицинское страхование, составляют 30 % от полной заработной платы по проекту, т.е. Ссоц. = Сзп*0,3.

Итак, в нашем случае:

$$С_{соц} = 121\,795{,}7 \cdot 0{,}3 = 36538{,}72 \text{ руб.}$$

4.5.4 Расчет затрат на электроэнергию

Данный вид расходов включает в себя затраты на электроэнергию, потраченную в ходе выполнения проекта на работу используемого оборудования, рассчитываемые по формуле:

$$С_{эл.об} = Р_{об} \cdot t_{об} \cdot ЦЭ, \quad (35)$$

где $Р_{об}$– мощность, потребляемая оборудованием, кВт;

ЦЭ – тариф на 1 кВт·час;

$t_{об}$ – время работы оборудования, час.

Для ТПУ ЦЭ = 5,748 руб./квт·час (с НДС).

Время работы оборудования вычисляется на основе итоговых данных таблицы – 9 для студента (ТРД) из расчета, что продолжительность рабочего дня равна 8 часов.

$$t_{об} = Т_{рд} \cdot К_t, \quad (36)$$

где$K_t \leq 1$– коэффициент использования оборудования по времени. Возьмем его равным 1.

Мощность, потребляемая оборудованием, определяется по формуле:

$$Р_{об} = Р_{ном} \cdot КС, \quad (37)$$

где $Р_{ном}$ – номинальная мощность оборудования, кВт;

$K_C \leq 1$ – коэффициент загрузки, зависящий от средней степени использования номинальной мощности. Для технологического оборудования малой мощности КС = 1.

Пример расчета затраты на электроэнергию для технологических целей приведен в таблице 11.

Таблица 11 – Затраты на электроэнергию технологическую

Наименование оборудования	Время работы оборудования $t_{ОБ}$,час	Потребляемая мощность $P_{ОБ}$, кВт	Затраты $Э_{ОБ}$, руб.
Персональный компьютер	464	0,3	800,12
Струйный принтер	2	0,1	1,15
Итого:			**801,27**

4.5.5 Расчет амортизационных расходов

В статье «Амортизационные отчисления» рассчитывается амортизация используемого оборудования за время выполнения проекта.

Используется формула:

$$С_{АМ} = \frac{Н_А \cdot Ц_{ОБ} \cdot t_{рф} \cdot n}{F_Д}, \quad (38)$$

где $Н_А$ – годовая норма амортизации единицы оборудования;

$Ц_{ОБ}$ – балансовая стоимость единицы оборудования с учетом ТЗР;

$t_{рф}$ – фактическое время работы оборудования в ходе выполнения проекта, учитывается исполнителем проекта;

n – число задействованных однотипных единиц оборудования.

Например, для ПК в 2019 г. (299 рабочих дней при шестидневной рабочей неделе) можно принять $F_Д = 299 \cdot 8 = 2392$ часа.

Для принтера из справочника $F_Д = 500$ часов.

При использовании нескольких типов оборудования расчет по формуле делается соответствующее число раз, затем результаты суммируются.

Для ПК найдем $Н_А = 0{,}4$. Для принтера $Н_А = 0{,}5$.

Стоимость ПК = 20 000 рублей. Время использования 304 часа, тогда для него:

$$С_{АМ}(ПК) = \frac{0{,}4 \cdot 20\,000 \cdot 464 \cdot 1}{2392} = 1551{,}84 \text{ руб.}$$

Стоимость принтера 5000 руб. Время использования 2 часа, тогда для него:

$$С_{АМ}(ПР) = \frac{0{,}5 \cdot 5\,000 \cdot 2 \cdot 1}{500} = 10 \text{ руб.}$$

Итого начислено амортизации 1 561,84 руб.

4.5.6 Расчет прочих расходов

В статье «Прочие расходы» отражены расходы на выполнение проекта, которые не учтены в предыдущих статьях, их следует принять равными 10% от суммы всех предыдущих расходов, т.е.

$$С_{проч} = (С_{мат} + С_{зд} + С_{соц} + С_{эл.об} + С_{ам}) * 0{,}1, \quad (39)$$

Для нашего примера это:

$$С_{проч} = (3\,990 + 121\,795{,}73 + 36\,538{,}72 + 801{,}12 + 1\,561{,}84) * 0{,}1 =$$
$$= 16\,468{,}74 \text{ руб.}$$

4.5.7 Расчет общей себестоимости разработки

Проведя расчет по всем статьям сметы затрат на разработку, можно определить общую себестоимость проекта. Данные результаты можно посмотреть в таблице 12.

Таблица 12 – Смета затрат на разработку проекта

Статья затрат	Условное обозначение	Сумма, руб.
Материалы и покупные изделия	$С_{мат}$	3 990
Основная заработная плата	$С_{зп}$	121 795,73

Продолжение таблицы 12

Статья затрат	Условное обозначение	Сумма, руб.
Отчисления в социальные фонды	$C_{соц}$	36 538,72
Расходы на электроэнергию	$C_{эл.}$	801,12
Амортизационные отчисления	$C_{ам}$	1 561,84
Прочие расходы	$C_{проч}$	16 468,74
Итого:		**181 156,15**

Таким образом, затраты на разработку составили C = 181 156,15руб.

4.5.8 Расчет прибыли

Прибыль примем в размере 10 % от полной себестоимости проекта. В нашем примере она составляет 18 115,61 руб. (10 %) от расходов на разработку проекта.

4.5.9 Расчет НДС

НДС составляет 20% от суммы затрат на разработку и прибыли. В нашем случае:

$$НДС = (181\ 156{,}15 + 18\ 115{,}61) * 0{,}2 = 39\ 854{,}35 \text{ руб.}$$

4.5.10 Цена разработки НИР

Цена равна сумме полной себестоимости, прибыли и НДС:

$$Ц_{НИР(КР)} = 181\ 156{,}15 + 18\ 115{,}61 + 39\ 854{,}35 = 239\ 126{,}11 \text{ руб.}$$

4.6 Оценка научно-технического эффекта

Социально-научный эффект проявляется в росте числа открытий, изобретений, увеличении суммарного объема научно-технической информации, полученной в результате выполнения выпускной квалификационной работы,

создании научного «задела», являющегося необходимой предпосылкой для проведения в будущем прикладных исследований и выполнения работа по модернизации конструкций выпускаемых изделий.

За последние годы появились предложения не только по качественной характеристике социального эффекта, но и по системе количественных показателей.

Элементом количественной оценки социально-научного эффекта следует считать определение научно-технического эффекта бакалаврской работы по следующей методике. Сущность этой методики состоит в том, что на основе оценок признаков работы определяется коэффициент научно-технического эффекта ВКР:

$$H_T = \sum_{i=1}^{3} r_i \cdot k_i \qquad (40)$$

где r_i – весовой коэффициент i-го признака (определяющийся по таблице 13); k_i – количественная оценка i-го признака.

Проведем расчет коэффициента научно-технического эффекта ВКР для алгоритма синонимизации текста.

Таблица 13 – Определение весового коэффициента

Признак научно-технического эффекта ВКР(i)	Применение значения весового коэффициента (r)
Уровень новизны	0,35
Теоретический уровень	0,25
Возможность реализации	0,4

Количественная оценка уровня новизны ВКР определяется на основе значений таблицы 14.

Таблица 14 – Количественная оценка уровня новизны ВКР

Уровень новизны разработки	**Характеристика уровня новизны**	**Баллы**
Принципиально новая	Результаты исследований открывают новое направление в данной области науки и техники	8-10
Новая	По-новому или впервые объяснены известные факты, закономерности	5-7
Относительно новая	Результаты исследований систематизируют и обобщают имеющиеся сведения, определяют пути дальнейших исследований	2-4
Уровень новизны разработки	**Характеристика уровня новизны**	**Баллы**
Традиционная	Работа выполнена по традиционной методике, результаты исследования носят информационный характер	1
Не обладающая новизной	Получен результат, который ранее был известен	0

Для данной выпускной квалификационной работы уровень новизны – относительно новая, баллы – 4.

Теоретический уровень полученных результатов выпускной квалификационной работы определяется на основе значения баллов, приведенных в таблице 15.

Таблица 15 – Теоретический уровень полученных результатов в ВКР

Теоретический уровень полученных результатов	Баллы
Установления закона, разработка новой теории	10
Глубокая разработка проблемы: многоаспектный анализ связей, взаимозависимости между фактами с наличием объяснения	8
Разработка способа (алгоритм, программ мероприятий, устройство, и т.д.)	6

Продолжение таблицы 15

Теоретический уровень полученных результатов	Баллы
Элементарный анализ связей между фактами с наличием гипотезы, симплексного прогноза, классификации, объясняющей версии, или практических рекомендаций частного характера	2
Описание отдельных элементарных фактов (вещей, свойств, отношений); изложение опыта, наблюдений, результатов измерений	0,5

В данной выпускной квалификационной работе был разработан алгоритм синонимизации текста, следовательно, теоретический уровень полученных результатов равен 6 баллам.

Возможность реализации научных результатов определяется на основе значения баллов из таблицы 16.

Таблица 16 – Время и масштабы реализации проекта

Время реализации	**Баллы**
В течение первых лет	10
От 5до 10 лет	4
Более 10 лет	2
Масштабы реализации	**Баллы**
Одно или несколько предприятий	2
Отрасль(министерство)	4
Народное хозяйство	10
Примечание: Баллы по времени и масштабам реализации складываются	

Способ синонимизации текста можно реализовать в течение первых лет (10 баллов), однако реализовать его можно только на отрасль (4 балла).

Рассчитаем коэффициент научно-технического эффекта:

$Нт = 0,35 \cdot 4 + 0,25 \cdot 6 + 0,4 \cdot 14 = 8,5$

Приведем таблицу оценок уровня научно-технического эффекта.

Таблица 17 – Оценка уровня научно-технического эффекта

Уровень научно-технического эффекта	Коэффициент научно-технического эффекта
Низкий	1-4
Средний	5-7
Сравнительно высокий	8-10
Высокий	11-14

В соответствии с таблицей 17, уровень научно-технического эффекта – сравнительно высокий.

Вывод по разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»

1. Проведено планирование НИР, а именно: определена структура и календарный план работы, трудоемкость, составлена ленточная диаграмма Ганта, и определен бюджет научно-исследовательской работы. В ходе планирования научно-исследовательских работ определён перечень работ, выполняемый рабочей группой. В данном случае рабочая группа состоит из двух человек: руководитель и инженер. Результаты соответствуют требованиям ВКР по срокам и иным параметрам.
2. Бюджет научно-технического исследования составил 239 126 руб. Бюджет НТИ состоит из материальных затрат (3 990 рублей), амортизационных отчислений (1 561 рублей), затрат на оплаты труда (121 795 рублей), отчислений во внебюджетные фонды (36 538 рубля) и накладных расходов (16 468 рубля).
3. Алгоритм синонимизации текста по многим показателям является более предпочтительным, чем другие варианты со значением 3,9.

4. В ходе выполнения раздела «Финансовый менеджмент» с помощью SWOT-анализа были выведены наиболее эффективные в сложившейся ситуации стратегии. После формирования бюджета затрат на проектирование суммарные капиталовложения составили 239 126,11 рублей. Уровень научно-технического эффекта – сравнительно высокий. Проект экономически целесообразен.

5. Капиталовложения в размере 239 126,11 рублей позволят реализовать разработанный проект по написанию алгоритма для синонимизации текста.

5 Социальная ответственность

С развитием научно-технического прогресса компьютеры находят все большее применение на производстве, в научно-исследовательских работах и в образовании. Однако, такое повсеместное применение вычислительной техники влечет за собой ряд различных заболеваний человека.

Так, для предупреждения вредного воздействия и сохранения здоровья сотрудника, работающего за компьютером, предусмотрен ряд мер по обеспечению безопасности трудовой деятельности.

В данной выпускной квалификационной работе разрабатывается алгоритм для генерации синонимичного текста на основе нейросетевой модели. Полученный алгоритм mнаправлен на снижение временных и финансовых затрат для создания уникальных текстов в сфере интернет маркетинга и SEO.

Так как исследование реализовано с помощью ЭВМ, то целью данного раздела является анализ соблюдения санитарных норм и правил в процессе работы над проектом с применением компьютера. Рассматриваются меры по защите сотрудника от негативного воздействия среды. Исследуются вредные и опасные факторы пагубно влияющих на здоровье человека при работе с компьютерами. Изучаются способы снижения воздействия вредных факторов до допустимых пределов. А также, рассматриваются возможные чрезвычайные ситуации и действия, которые офисный работник должен выполнить в случае возникновения ЧС.

В рамках данной работы объектом исследования раздела «Социальная ответственность» является алгоритм, реализуемый в виде программного приложения с помощью ЭВМ. В связи с чем необходимо знать вредные и опасные факторы при работе с ЭВМ, а также способы их устранения.

5.1 Правовые и организационные вопросы обеспечения безопасности

Рабочее место должно быть организовано в соответствии ГОСТ 12.2.032-78 «Рабочее место при выполнении работ сидя. Общие эргономические требования» [16] и СанПиН 2.2.2/2.4.1340-03 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы» [17]. При проектировании рабочих мест должны быть учтены освещенность, шум, температура, влажность, наличие электромагнитных полей и другие санитарно-гигиенические требования.

Основные требования СанПиН 2.2.2/2.4.1340-03 [17]:

– При размещении рабочих мест с персональными электронно-вычислительными машинами (ПЭВМ) расстояние между рабочими столами с дисплеями должно быть не менее 2,0 м, а расстояние между боковыми поверхностями дисплеев – не менее 1,2 м.

– Экран дисплея должен находиться от глаз пользователя на расстоянии 0,6 – 0,7 м, но не ближе 0,5 м.

– Конструкция рабочего стола должна обеспечивать оптимальное размещение на рабочей поверхности используемого оборудования с учетом его количества и конструктивных особенностей, характера выполняемой работы. При этом допускается использование рабочих столов различных конструкций, отвечающих современным требованиям эргономики.

– Конструкция рабочего стула должна обеспечивать поддержание рациональной рабочей позы при работе на ПЭВМ позволять изменять позу с целью снижения статического напряжения мышц шейно-плечевой области и спины для предупреждения развития утомления. Тип рабочего стула следует выбирать с учетом роста пользователя, характера и продолжительности работы с ПЭВМ.

– Рабочий стул должен быть подъемно-поворотным, регулируемым по высоте и углам наклона сиденья и спинки, а также расстоянию спинки от

переднего края сиденья, при этом регулировка каждого параметра должна быть независимой, легко осуществляемой и иметь надежную фиксацию.

– Поверхность сиденья, спинки и других элементов стула должна быть полумягкой, с нескользящим, слабо электризующимся и воздухопроницаемым покрытием, обеспечивающим легкую очистку от загрязнений.

– Допустимый уровень напряженности электромагнитного поля в 0,5 метрах относительно дисплея и системного – 2,5 [*В/м*];

– Допустимый уровень плотности магнитного потока в 0,5 метрах относительно дисплея и системного блока в диапазоне частот 5-2 [*КГц*] составляет 250 [*нТл*]; поверхностный электростатический потенциал составляет 500 [*В*].

В соответствии с ГОСТ 12.2.032-78 «Рабочее место при выполнении работ сидя. Общие эргономические требования» продолжительность рабочего дня не должна превышать восьми часов, при этом каждый час необходим перерыв продолжительностью 15 минут [16].

Нормативно-правовая база обеспечения безопасности жизнедеятельности населения и защиты территорий регламентирует обязанности и права государственных органов, общественных организаций, должностных лиц и всех граждан, закрепляет и регулирует устройство и назначение специальных органов управления в области защиты от ЧС, определяет ответственность всех уровней власти и граждан

Правовой основой законодательства в области обеспечения безопасности жизнедеятельности является Конституция.

1. Закон РФ «О защите населения и территорий от чрезвычайных ситуаций природного и техногенного характера» № 68 – ФЗ от 21.12. 1994 г. [18] определяет общие для РФ организационно-правовые нормы в области защиты граждан РФ, иностранных граждан и лиц без гражданства, находящихся на территории РФ, всего земельного, водного, воздушного пространства в пределах РФ или его части, объектов производственного и

социального назначения, а также окружающей природной среды от ЧС природного и техногенного характера.

2. Закон РФ «О гражданской обороне» № 28 – ФЗ от 12.02.1998 г. [19] определяет задачи в области гражданской обороны и правовые основы их осуществления, полномочия органов государственной власти РФ, органов исполнительной власти субъектов РФ, органов местного самоуправления, организаций независимо от их организационно-правовых форм и форм собственности, а также силы и средства гражданской обороны.

Обеспечение экологической безопасности на территории РФ, формирование и укрепление экологического правопорядка основаны на действии Федерального закона «Об охране окружающей природной среды.

5.2 Производственная безопасность

Производственная безопасность – система организационных мероприятий и технических средств, предотвращающих или уменьшающих вероятность воздействия на работающих опасных травмирующих производственных факторов, возникающих в рабочей зоне в процессе трудовой деятельности.

Согласно ГОСТ 12.0.003-2015 «Опасные и вредные производственные факторы. Классификация» [20] неблагоприятные производственные факторы по результирующему воздействию на организм работающего человека подразделяют на:

– вредные производственные факторы, то есть факторы, приводящие к заболеванию, в том числе усугубляющие уже имеющиеся заболевания;

– опасные производственные факторы, то есть факторы, приводящие к травме, в том числе смертельной.

Перечень опасных и вредных факторов, характерных для проектируемой производственной среды представлен в виде таблицы 18.

Таблица 18– Возможные опасные и вредные факторы

Факторы (ГОСТ 12.0.003-2015)	Этапы работ		Нормативные документы
	Разработка	Эксплуатация	
1. Недостаточная освещенность рабочей зоны	+	+	СП 52.13330.2016 Естественное и искусственное освещение. Актуализированная редакция СНиП 2305-95* [12]
2. Превышение уровня шума	+	+	СН 2.2.4/2.1.8.562–96. Шум на рабочих местах, в помещениях жилых, общественных зданий и на территории застройки [18]
3. Отклонение показателей микроклимата	+	+	СанПиН 2.2.4.548–96. Гигиенические требования к микроклимату производственных помещений [1]
4. Повышенное значение напряжения в электрической цепи	+	+	СанПиН 2.2.2/2.4.1340–03 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы» [5]

5.2.1 Анализ выявленных опасных и вредных факторов

Недостаточная освещенность рабочей зоны. Неправильно организованное освещение может негативно сказаться на здоровье работников, может ухудшиться зрение. Также недостаточная освещенность может привести к быстрому утомлению и снижению работоспособности. В соответствии со сводом нормативных актов СП 52.13330.2016 «Естественное и искусственное освещение. Актуализированная редакция СНиП 23-05-95*» освещенность при разработке алгоритма должна составлять 300-500 лк. Освещение не должно создавать бликов на поверхности экрана. Освещенность поверхности экрана не должна превышать 300 лк. Коэффициент пульсации не должен превышать 5% [21].

Для искусственного освещения помещений с персональными компьютерами следует применять светильники типа ЛПО36 с зеркализованными решетками, укомплектованные высокочастотными пускорегулирующими аппаратами. Допускается применять светильники прямого света, преимущественно отраженного света типа ЛПО13, ЛПО5, ЛСО4, ЛПО34, ЛПО31 с люминесцентными лампами типа ЛБ. Допускается применение светильников местного освещения с лампами накаливания. Светильники должны располагаться в виде сплошных или прерывистых линий сбоку от рабочих мест параллельно линии зрения пользователя при разном расположении компьютеров. При периметральном расположении линии светильников должны располагаться локализовано над рабочим столом ближе к его переднему краю, обращенному к оператору. Защитный угол светильников должен быть не менее 40 градусов. Светильники местного освещения должны иметь не просвечивающийся отражатель с защитным углом не менее 40 градусов.

Превышение уровня шума**.** При разработке программного обеспечения, основными источниками шума являются:

- Вентиляторы и кулеры системных блоков, находящихся в комнате;
- Жесткие диски и системные блоки.

Также могут иметься иные источники шума, находящиеся за пределами рабочего помещения (строительные и ремонтные работы, массовые мероприятия и т.д.).

В соответствии с СН 2.2.4/2.1.8.562–96. «Шум на рабочих местах, в помещениях жилых, общественных зданий и на территории застройки» при выполнении основной работы на ПЭВМ уровень шума на рабочем месте не должен превышать 50 дБА. Допустимые значения уровней звукового давления в октавных полосах частот и уровня звука, создаваемого ПЭВМ приведены в таблице 19 [22]

Таблица 19 – Допустимые значения уровней звукового давления в октавных полосах частот и уровня звука, создаваемого ПЭВМ

Уровни звукового давления в октавных полосах со среднегеометрическими частотами									Уровни звука в дБА
31,5 Гц	63 Гц	125 Гц	250 Гц	500 Гц	1000 Гц	2000 Гц	4000 Гц	8000 Гц	
86 дБ	71 дБ	61 дБ	54 дБ	49 дБ	45 дБ	42 дБ	40 дБ	38 дБ	50

В случае несоответствия показателей шума установленным нормам, необходимо прибегнуть к мерам по их оптимизации:

- замена компонент ЭВМ на менее шумные аналоги;
- установка звуконепроницаемых окон и дверей.

Отклонение показателей микроклимата. При определенных значениях параметров микроклимата человек испытывает состояние теплового комфорта, что способствует повышению производительности труда, предупреждению простудных заболеваний. Неблагоприятные значения микроклиматических показателей могут стать причиной снижения производственных показателей в работе, привести к таким заболеваниям, как различные формы простуды, радикулит, тонзиллит, хронический бронхит и др. Слишком высокая влажность затрудняет терморегуляцию, а слишком низкая вызывает пересыхание слизистых: дыхательных путей и глаз. Также уровень влажности влияет на электростатические и электромагнитные поля: чем он выше, тем слабее влияние указанных полей.

Оптимальные микроклиматические условия установлены по критериям оптимального теплового и функционального состояния человека. В соответствии с СанПиН 2.2.4.548–96. «Гигиенические требования к микроклимату производственных помещений» работа программиста соответствует категории работ *Ia* (работы с интенсивностью энергозатрат до 120 ккал/ч (до 139 Вт), производимые сидя и сопровождающиеся незначительным физическим напряжением) [23]. Оптимальные параметры

микроклимата для категории работ *Ia* приведены в таблице 20 [23], допустимые параметры микроклимата – в таблице 21.

Таблица 20 – Оптимальные параметры микроклимата

Сезон	Температура воздуха, *t* [*°C*]	Температура поверхностей, *t* [*°C*]	Относительная влажность, %	Скорость движения воздуха, [*м/с*]
Холодный и переходный (среднесуточная температура меньше 10*°C*)	22-24	21 – 25	40-60	0,1
Теплый (среднесуточная температура воздуха 10*°C* и выше)	23-25	22-26	40-60	0,1

Таблица 21 – Допустимые параметры микроклимата

Период года	Температура воздуха, *°C*		Температура поверхностей, °С	Относительная влажность воздуха, %	Скорость движения воздуха, м/с	
	Диапазон ниже оптимальных величин	Диапазон выше оптимальных величин			Ниже оптим. величин не более	Выше оптимальных величин не более
Холодный переходный	20,0-21,9	24,1-25,0	19,0 -26,0	15-75	0,1	0,1
Теплый	21,0-22,9	25,1-28,0	20,0 -29,0	15-75	0,1	0,2

В случае несоответствия показателей микроклимата установленным нормам необходимо прибегнуть к мерам по их оптимизации:

– установка кондиционеров и обогревателей в рабочих помещениях;

– усовершенствование, ремонт вентиляционной и отопительной систем.

Поражение человеческого организма электрическим током может служить причиной травм различного характера: повреждение мышечного и кожного покровов, ожоги различной степени и т.д. Последствия действия тока на организм человека зависят от силы тока, длительности его действия, пути тока в теле и индивидуальных свойств организма.

Для предотвращения поражения электрическим током при работе с компьютером необходимо:

– обеспечить недоступность токоведущих частей для прикосновения;

– подключать все электрические приборы, включая ЭВМ, к сети питания только через сетевой фильтр;

– избегать возникновения повышенной влажности;

– не снимать боковую крышку корпуса ЭВМ при включённой сети питания;

– обеспечивать чистоту помещения и не допускать запыленности воздуха.

Часто в процессе эксплуатации ЭВМ возникает необходимость замены и ремонта ее составляющих. В соответствии с СанПиН 2.2.2/2.4.1340-03 запрещено проводить ремонт ЭВМ непосредственно в рабочих, лабораторных и рабочих помещениях [17].

5.3 Экологическая безопасность

5.3.1 Анализ влияния объекта исследования на окружающую среду

Объект исследования является теоретическим и не оказывает влияния на окружающую среду.

5.3.2 Анализ влияния процесса исследования на окружающую среду

В ходе данной работы были использованы следующие ресурсы:

– электроэнергия для работы компьютера;

– бумага;

– люминесцентные лампы.

С точки зрения потребления ресурсов компьютер потребляет сравнительно небольшое количество электроэнергии, что положительным образом сказывается на общей экономии потребления электроэнергии в целом.

При написании ВКР вредных выбросов в атмосферу, почву и водные источники не производилось, радиационного заражения не произошло, чрезвычайные ситуации не наблюдались, поэтому не оказывались существенные воздействия на окружающую среду, и никакого ущерба окружающей среде не было нанесено.

5.3.3 Обоснование мероприятий по защите окружающей среды

В связи с тем, что огромная масса информации содержится на бумажных носителях, уничтожение бумаги играет очень важную роль. Среди основных методов уничтожения, которые применяются на сегодняшний день для бумажных документов, следует отметить следующие:

– Сжигание документов;

– Шредирование;

– Закапывание;

– Химическая обработка.

Переработка оргтехники включает в себя несколько этапов:

Первый этап – удаление всех опасных компонентов.

Второй этап – удаление всех крупных пластиковых частей. В большинстве случаев эта операция также осуществляется вручную. Оставшиеся после разборки части отправляют в большой измельчитель, и все дальнейшие операции автоматизированы.

Третий этап – измельченные в гранулы остатки компьютеров подвергаются сортировке. Сначала с помощью магнитов извлекаются все

железные части. Затем приступают к выделению цветных металлов, которых в ПК значительно больше.

Перегоревшие люминесцентные лампы можно отнести в свой районный ДЕЗ или РЭУ, где установлены специальные контейнеры. Там их должны бесплатно принять.

5.4 Безопасность в чрезвычайных ситуациях

5.4.1 Анализ вероятных ЧС, которые может инициировать объект исследований

Объект исследования является теоретическим и не может привести к возникновению ЧС.

5.4.2 Анализ вероятных ЧС, которые могут возникнуть на рабочем месте при проведении исследований

Наиболее вероятной чрезвычайной ситуацией при написании выпускной квалификационный является пожар на рабочем месте.

В качестве противопожарных мероприятий должны быть применены следующие меры:

– В помещении должны находится средства тушения пожара;

– Электрическая проводка электрооборудования должна быть исправна;

– Все сотрудники должны знать место нахождения средств пожаротушения и уметь ими воспользоваться, средств связи и номера экстренных служб.

Рабочее помещение оборудовано в соответствии с требованиями пожарной безопасности. Имеется порошковый огнетушитель, а также пожарная сигнализация и средства связи.

5.4.3 Обоснование мероприятий по предотвращению ЧС и разработка порядка действия в случае возникновения ЧС

В случае возникновения пожара сообщить о нем руководителю и постараться устранить очаг возгорания имеющимися силами при помощи первичных средств пожаротушения. Привести в действие ручной пожарный извещатель, если очаг возгорания потушить не удается, а также сообщить о возгорании в службу пожарной охраны по телефону 101 или 112, сообщить адрес, место и причину возникновения пожара.

Выводы по разделу «Социальная ответственность»

Проанализировав и оценив условия труда в рабочем помещении, где была разработана выпускная квалификационная работа, можно сделать выводы, что грубых нарушений по организации работы не обнаружено и нормы безопасности соблюдены. Само помещение и рабочее место удовлетворяет всем нормативным требованиям. Действие вредных и опасных факторов сведено к минимуму, т.е. микроклимат, освещение и электробезопасность соответствуют требованиям, предъявленным в соответствующих нормативных документах. Не стоит забывать, что монитор компьютера служит источником вредного фактора и отрицательно влияет на здоровье офисного сотрудника. Во избежание этого, нужно делать перерывы в работе и проводить специальные комплексы упражнений для разминки тела.

Заключение

В ходе выполнения выпускной квалификационной работы была достигнута основная цель работы – разработан алгоритм для генерации текста на основе нейросетевой модели. Основой для создания является язык программирования Python, а также следующие библиотеки: NumPy, SciPy, Transformers, pymorphy2, torch.

Данная разработка позволит автоматизировать процесс создания различного текстового контента для различных интернет ресурсов.

В процессе разработки были выполнены следующие задачи:

- обзор литературы в выбранной предметной области;
- разработка алгоритм для синонимизации текста;
- разработка приложения для синонимизации текста;
- произведено сравнение данного алгоритма с различными нейросетевыми моделями.

Данная разработка подлежит дальнейшему развитию и улучшению для более качественной генерации синонимичных текстов. Для улучшения качества работы алгоритма можно провести эксперимент со следующими изменениями:

- замена базовой модели BERT на более продуктивную модификацию;
- расширение количества словарей;
- использование более сложного алгоритма для определения конечной выборки (например, лучевого поиска).

Список использованных источников

[1] Кочеткова Н.А. Статистические языковые методы. Коллокации и коллигации / Языкознание и литературоведение [Электронный ресурс]. - 2013. - Режим доступа: https://cyberleninka.ru, свободный. – Загл. с экрана. (дата обращения: 10.03.2020).

[2] Cristopher D. Manning. Foundations of statistical natural language processing / Cristopher D. Manning, Hinrich Schutze. // Massachusetts Institute of Technology. – 1999. – 680 p.

[3] Yaov Goldberg. Neural Network Methods for Natural Language Processing, 2017 – P. 947-4040.

[4] Dan Jurafsky Speech and Language Processing / Dan Jurafsky, James H. – 2019 – 548 p.

[5] Transformer-XL Explained: Combining Transformers and RNNs into a State-of-the-art Language Model [Электронный ресурс].- Режим доступа: https://towardsdatascience.com/Transformer-xl-explained-combining-Transformers-and-rnns-into-a-state-of-the-art-language-model-c0cfe9e5a924, свободный. – Загл. с экрана.

[6] Elman J. L. Finding structure in time / Elman J. L. – 1990. – Vol. 14, №2. – P. 179–211.

[7] Hochreiter S. Long short-term memory. Neural computation / Hochreiter S., Schmidhuber J. – 1997 – Vol. 9, №8. – P. 1735–1780.

[8] K. Cho. On the properties of neural machine translation: Encoder-decoder approaches. [Электронный ресурс]. - Режим доступа: https://arxiv.org/pdf/1409.1259.pdf, свободный. - 2013. – Загл. с экрана. (дата обращения: 15.05.2020)

[9] J. Chung. Empirical evaluation of gated recurrent neural networks on sequence modeling. [Электронный ресурс]. - 2013. - Режим доступа: https://arxiv.org/pdf/1412.3555.pdf, свободный. – 2014 – Загл. с экрана. (дата обращения: 15.05.2020)

[10] The Illustrated Transformer [Электронный ресурс].- Режим доступа: http://jalammar.github.io/illustrated-Transformer/, свободный. – Загл. с экрана. (дата обращения: 10.02.2020)

[11] Language Models with Transformers [Электронный ресурс].- Режим доступа: https://arxiv.org/pdf/1904.09408.pdf, свободный. – Загл. с экрана. (дата обращения: 10.02.2020)

[12] Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing [Электронный ресурс].- Режим доступа: https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html, свободный. – Загл. с экрана. (дата обращения: 10.02.2020)

[13] BERT [Электронный ресурс].- Режим доступа: https://github.com/google-research/bert, свободный. – Загл. с экрана. (дата обращения: 15.04.2020)

[14] The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning) [Электронный ресурс].- Режим доступа: https://github.com/google-research/bert, свободный. – Загл. с экрана. (дата обращения: 15.04.2020)

[15] Rico Sennrich. Neural Machine Translation of Rare Words with Subword Units / Rico Sennrich, Barry Haddow. [Электронный ресурс]. - Режим доступа: https://arxiv.org/pdf/1508.07909.pdf, свободный. – Загл. с экрана. (дата обращения: 20.05.2020)

[16] ГОСТ 12.2.032-78 «ССБТ. Рабочее место при выполнении работ сидя. Общие эргономические требования».

[17] СанПиН 2.2.2/2.4.1340-03 Гигиенические требования к персональным электронно-вычислительным машинам и организации работы

[18] Федеральный закон от 21 декабря 1994 г. № 68-ФЗ. О защите населения и территорий от чрезвычайных ситуаций природного и техногенного характера (с изменениями и дополнениями).

[19] Федеральный закон от 12 февраля 1998 г. № 28 – ФЗ О гражданской обороне (с изменениями и дополнениями).

[20] ГОСТ 12.0.003-2015 ССБТ. Опасные и вредные производственные факторы. Классификация.

[21] СП 52.13330.2016 Естественное и искусственное освещение. Актуализированная редакция СНиП 23-05-95*

[22] 2.2.4/2.1.8.562–96. Шум на рабочих местах, в помещениях жилых, общественных зданий и на территории застройки

[23] СанПиН 2.2.4.548–96. Гигиенические требования к микроклимату производственных помещений.

[24] Трудовой кодекс Российской Федерации от 30.12.2001 N 197-ФЗ (ред. от 27.12.2018)

[25] ГОСТ Р 50923-96. «Дисплеи. Рабочее место оператора. Общие эргономические требования и требования к производственной среде. Методы измерения».

[26] СП 52.13330.2011 Свод правил естественное и искусственное освещение.

[27] ГОСТ 12.1.038–82 «Система стандартов безопасности труда. Электробезопасность. Предельно допустимые значения напряжений прикосновения и токов».

[28] ГОСТ 12.1.005-88 «Система стандартов безопасности труда. Общие санитарно-гигиенические требования к воздуху рабочей зоны».

[29] СанПиН 2.2.1/2.1.1.1278-03 Гигиенические требования к естественному, искусственному и совмещенному освещению жилых и общественных зданий.

[30] СанПиН 2.2.4/2.1.8.10-32-2002 Шум на рабочих местах, в помещениях жилых, общественных зданий и на территории жилой застройки.

Приложение А

(рекомендуемое)

Листинг программы для получения распределения вероятностей от нейросетевой модели BERT.

```
import torch
from Transformers import AutoTokenizer, BertTokenizer, BertForMaskedLM, AutoModelWithLMHead
import logging
import openpyxl
import gensim
from pandas import DataFrame, ExcelWriter
import spacy
import pymorphy2
import udpipe
from datetime import datetime

from dictionary_top import Dictionary, PredictionsForWord, PredictionsForText

logging.config.fileConfig('log_config.conf')
logger = logging.getLogger("graduateWork")

logger.info("Program started")

dictionaries = ["cost.txt", "material.txt", "positive.txt", "usability.txt"]
for path in dictionaries:
   PredictionsForWord.add_vocabulary(path)

modelpath = "DrMatters/rubert_cased"
tokenizer = BertTokenizer.from_pretrained(modelpath)
model = AutoModelWithLMHead.from_pretrained(modelpath)
model.eval()

text = "он подарил мне блестящий меч"
my_predictions = PredictionsForText(text)
topn = 30

splitted_text = list(())
```

```
nlp = spacy.load("ru2")
doc = nlp(text)
splitted_text.append("[CLS]")
for token in doc:
    splitted_text.append(token.text)

final_dict = dict()

for word_index, word in enumerate(splitted_text):
    if word == "[SEP]" or word == "[CLS]":
        continue
    splitted_text[word_index] = '[MASK]'

    masked_index = None
    tokenized_text = list(())

    for word2 in splitted_text:
        tokenized_word = tokenizer.tokenize(word2)
        tokenized_text.extend(tokenized_word)
        if word2 == '[MASK]':
            masked_index = len(tokenized_text) - 1

    tokens_tensor = torch.tensor([indexed_tokens])
    predictions = model(tokens_tensor)[0]
    top_100 = list(())

    for value, ids in zip(*torch.topk(predictions[0, masked_index], topn, largest=True)):
        predicted_token = tokenizer.convert_ids_to_tokens([ids.item()])
        top_100.append((predicted_token, value.item()))
        my_predictions.add_bert(word, predicted_token[0], value.item(), tokenizer)

    final_dict[word] = top_100
    splitted_text[word_index] = word

taiga_fasttext = gensim.models.KeyedVectors.load("taiga_fasttext/model.model")
taiga_skipgram = gensim.models.KeyedVectors.load_word2vec_format("taiga_skipgram/model.bin",
binary=True)
```

```
export_dict_fasttext = dict()
export_dict_skipgram = dict()

model = udpipe.init()
process_pipeline = udpipe.Pipeline(model, 'tokenize', udpipe.Pipeline.DEFAULT, udpipe.Pipeline.DEFAULT, 'conllu')
for word in doc:
    if str(word) == "[SEP]" or str(word) == "[CLS]":
        continue
    res = udpipe.unify_sym(str(word))
    preproced_word = udpipe.process(process_pipeline, text=res, keep_punct=True)[0]
    if word in taiga_fasttext:
        top_100 = list(())
        for i in taiga_fasttext.most_similar(positive=[str(word)], topn=topn):
            my_predictions.add_fasttext(str(word), i[0], i[1])
            top_100.append((i[0], i[1]))
        export_dict_fasttext[word] = top_100
    else:
        export_dict_fasttext[word] = "NOT FOUND"
    if preproced_word in taiga_skipgram:
        top_100 = list(())
        for i in taiga_skipgram.most_similar(positive=[preproced_word], topn=topn):
            my_predictions.add_skipgram(str(word), i[0], i[1])
            top_100.append((i[0], i[1]))
        export_dict_skipgram[preproced_word] = top_100
    else:
        export_dict_skipgram[preproced_word] = "NOT FOUND"

export_dict = dict()

for key, value in final_dict.items():
    export_dict[key] = value

writer = ExcelWriter(str(datetime.now().isoformat()) + 'output.xlsx', engine='xlsxwriter')
DataFrame(export_dict).to_excel(writer, sheet_name='Bert')
DataFrame(export_dict_fasttext).to_excel(writer, sheet_name='FastText')
DataFrame(export_dict_skipgram).to_excel(writer, sheet_name='Skipgram')
```

```
writer.save()
logger.info("Done!")
```

Приложение Б

(рекомендуемое)

Таблица Б.1 – Полученное распределение вероятностей

	«Он»		«подарил»		«мне»		«блестящий»		«меч»	
	Предположение	Оценка	Предположение	Оценка	Предположение	Оценка	Предположение	Оценка	Предположение	Оценка
BERT	Он	10,96	дал	11,24	ему	9,39	свой	11,32	вещи	8,97
	Бог	8,79	показал	10,79	ей	8,11	этот	10,07	волосы	8,27
	Господь	7,97	подарил	10,20	мне	7,38	его	10,05	цветы	8,26
	Аллах	7,74	показывает	10,03	им	7,03	мой	8,43	глаза	8,25
	Отец	7,68	послал	9,89	два	7,02	большой	8,35	украшения	7,69
skipgram	-	-	подарить_VERB	0,79	-	-	блестящий_VERB	0,94	клинок_NOUN	0,88
	-	-	подаренный_VERB	0,71	-	-	блестеть_ADJ	0,86	секира_NOUN	0,84
	-	-	подарить_NOUN	0,70	-	-	сверкать_ADJ	0,62	кинжал_NOUN	0,82
	-	-	подарилийский_NOUN	0,69	-	-	блестеть_VERB	0,61	копье_NOUN	0,80
	-	-	подарать_VERB	0,65	-	-	сверкающий_VERB	0,60	сабля_NOUN	0,79
fasttext	господь::иисус	0,66	дарительница	0,51	мне	0,80	блестевший	0,82	клинок	0,87
	господин::иисус	0,66	дариэл	0,51	мне,	0,80	сверкающий	0,81	кинжал	0,83
	господь::иисус::христос	0,63	дарио	0,50	мне,я	0,77	блескучий	0,76	секира	0,82
	бож	0,63	подарю	0,49	мнеь	0,77	серо-серебристый	0,76	копье	0,81
	господин::бог	0,63	дарис	0,49	мне,но	0,70	серебристый	0,75	-меч	0,79

Приложение В

(рекомендумое)

Листинг классов и методов используемых для последних трех этапов алгоритма.

```
from Transformers import BertTokenizer
import logging
import udpipe
import spacy
import pymorphy2
import numpy as np
from scipy.special import softmax
module_logger = logging.getLogger("graduateWork.dictionary_top")


class Dictionary:
    def __init__(self, path: str):
        """Constructor"""
        self._name = path
        file = open(path, 'r')
        dict = set()
        for line in file:
            dict.add(line.rstrip("\n"))
        self._dictionary = dict

    def __str__(self):
        stroka = "Словарь: " + self._name + "\n"
        for word in self._dictionary:
            stroka += word + "|"
        return stroka


class PredictionsForWord:
    """docstring"""
    _vocabulary = []
    _morph = None
```

```
    def __init__(self, word: str):
        """Constructor"""
        self._word = word
        self._bert = {}
        self._normalized_bert = {}
        self._skipgram = {}
        self._fasttext = {}
        self._bert_dictionary = {}
        self._skipgram_dictionary = {}
        self._fasttext_dictionary = {}
        if PredictionsForWord._morph == None:
            PredictionsForWord._morph = pymorphy2.MorphAnalyzer()



    def __str__(self):
        stroka = "----------------------------------------"
        stroka += "Словарь предсказания для слова: " + self._word + "\n"
        stroka += "Список словарей: \n"
        for dict in PredictionsForWord._vocabulary:
            stroka += str(dict) + "\n"
        stroka += "Предсказания бертом: \n" + str(self._bert) + "\n"
        stroka += "Предсказания skipgram: \n" + str(self._skipgram) + "\n"
        stroka += "Предсказания fasttext: \n" + str(self._fasttext) + "\n"
        stroka += "Включение слов из словарей в выдаче берта: \n" + str(self._bert_dictionary) + "\n"
        stroka += "Включение слов из словарей в выдаче skipgram: \n" + str(self._skipgram_dictionary) +
"\n"
        stroka += "Включение слов из словарей в выдаче fasttext: \n" + str(self._fasttext_dictionary) + "\n"
        stroka += "----------------------------------------"
        return stroka

    def get_bert(self, word: str):
        logger = logging.getLogger("graduateWork.dictionary_top.get_bert")
        cell = self._bert.get(word)
        if cell == None:
            logger.info("Word: %s not found in bert" % (cell))
        return cell
```

```
def get_normalized_bert(self, word: str):
    logger = logging.getLogger("graduateWork.dictionary_top.get_normalized_bert")
    cell = self._normalized_bert.get(word)
    if cell == None:
        logger.info("Word: %s not found in normalized bert" % (cell))
    return cell

def get_skipgram(self, word: str):
    logger = logging.getLogger("graduateWork.dictionary_top.get_skipgram")
    cell = self._skipgram.get(word)
    if cell == None:
        logger.info("Word: %s not found in skipgram" % (cell))
    return cell

def get_fasttext(self, word: str):
    logger = logging.getLogger("graduateWork.dictionary_top.get_fasttext")
    cell = self._fasttext.get(word)
    if cell == None:
        logger.info("Word: %s not found in fasttext" % (cell))
    return cell

def add_bert(self, predicted: str, score, tokenizer: BertTokenizer):
    self._bert[predicted] = score
    tokenized = tokenizer.tokenize(predicted)
    normal_form = PredictionsForWord.get_normal_form(tokenized[0])
    self._normalized_bert[normal_form] = predicted

def add_skipgram(self, predicted: str, score):
    self._skipgram[predicted] = score

def add_fasttext(self, predicted: str, score):
    self._fasttext[predicted] = score

def get_normal_form(word: str):
    morph_parse = PredictionsForWord._morph.parse(word)
    return morph_parse[0].normal_form

def find_bert(self, tokenizer: BertTokenizer):
```

```
        logger = logging.getLogger("graduateWork.dictionary_top.find_bert")
        for dictionary in PredictionsForWord._vocabulary:
            for dict_word in dictionary._dictionary:
                tokenized = tokenizer.tokenize(dict_word)
                if tokenized.__len__() > 1:
                    logger.info(
                        "Word: %s have more then one token. Tokens count: %d" % (dict_word,
tokenized.__len__()))
                normalized_token = PredictionsForWord.get_normal_form(tokenized[0])
                needed_bert_form = self.get_normalized_bert(normalized_token)
                if needed_bert_form == None:
                    logger.info("Word: %s doesnt have score in bert model" % (tokenized[0]))
                    continue
                score = self.get_bert(needed_bert_form)
                if score == None:
                    logger.info("Word: %s doesnt have score in bert model" % (tokenized[0]))
                    continue
                self._bert_dictionary[needed_bert_form] = (score, dictionary._name)

    def find_skipgram(self, tokenizer: udpipe.Pipeline):
        logger = logging.getLogger("graduateWork.dictionary_top.find_skipgram")
        for dictionary in PredictionsForWord._vocabulary:
            for dict_word in dictionary._dictionary:
                unified = udpipe.unify_sym(dict_word)
                tokenized = udpipe.process(tokenizer, text=unified, keep_punct=True)
                if tokenized == None:
                    logger.info("Word: %s doesnt have token in skipgram model" % (dict_word))
                    continue
                score = self.get_skipgram(tokenized[0])
                if score == None:
                    logger.info("Word: %s doesnt have score in skipgram model" % (tokenized[0]))
                    continue
                self._skipgram_dictionary[dict_word] = (score, dictionary._name)

    def find_fasttext(self):
        logger = logging.getLogger("graduateWork.dictionary_top.find_fasttext")
        for dictionary in PredictionsForWord._vocabulary:
            for dict_word in dictionary._dictionary:
```

```
            unified = udpipe.unify_sym(dict_word)
            score = self.get_fasttext(unified)
            if score == None:
                logger.info("Word: %s doesnt have score in fasttext model" % (unified))
                continue
            self._fasttext_dictionary[dict_word] = (score, dictionary._name)

def find_all(self, bert_tokenizer: BertTokenizer, udpipe: udpipe.Pipeline):
    self.find_bert(bert_tokenizer)
    self.find_skipgram(udpipe)
    self.find_fasttext()

def add_vocabulary(path: str):
    dict = Dictionary(path)
    PredictionsForWord._vocabulary.append(dict)

def sinonimize(self):
    tmp = {}
    for candidates in self._bert_dictionary:
        dict_name = self._bert_dictionary[candidates][1]
        value = tmp.get(dict_name, 0)
        tmp[dict_name] = value + 1
    max = 0
    leader = None
    for candidates in tmp:
        if tmp[candidates] > max:
            max = tmp[candidates]
            leader = candidates
    if leader == None:
        return self._word
    word = []
    scores = np.array([])
    for predicted_word in self._bert_dictionary:
        cell = self._bert_dictionary[predicted_word]
        if cell[1] == leader:
            word.append(predicted_word)
            scores = np.append(scores, [cell[0]])
```

```
    normalized = softmax(scores)
    print(word)
    print(normalized)
    predicted = np.random.choice(word, 1, p=normalized)
    print(predicted)
    return predicted[0]

def sinonimize_skipg(self):
    tmp = {}
    for candidates in self._skipgram_dictionary:
        dict_name = self._skipgram_dictionary[candidates][1]
        value = tmp.get(dict_name, 0)
        tmp[dict_name] = value + 1
    max = 0
    leader = None
    for candidates in tmp:
        if tmp[candidates] > max:
            max = tmp[candidates]
            leader = candidates
    if leader == None:
        return self._word
    word = []
    scores = np.array([])
    for predicted_word in self._skipgram_dictionary:
        cell = self._skipgram_dictionary[predicted_word]
        if cell[1] == leader:
            word.append(predicted_word)
            scores = np.append(scores, [cell[0]])

    normalized = softmax(scores)
    print(word)
    print(normalized)
    predicted = np.random.choice(word, 1, p=normalized)
    print(predicted)
    return predicted[0]

def sinonimize_fast(self):
    tmp = {}
```

```
        for candidates in self._fasttext_dictionary:
            dict_name = self._fasttext_dictionary[candidates][1]
            value = tmp.get(dict_name, 0)
            tmp[dict_name] = value + 1
        max = 0
        leader = None
        for candidates in tmp:
            if tmp[candidates] > max:
                max = tmp[candidates]
                leader = candidates
        if leader == None:
            return self._word
        word = []
        scores = np.array([])
        for predicted_word in self._fasttext_dictionary:
            cell = self._fasttext_dictionary[predicted_word]
            if cell[1] == leader:
                word.append(predicted_word)
                scores = np.append(scores, [cell[0]])

        normalized = softmax(scores)
        print(word)
        print(normalized)
        predicted = np.random.choice(word, 1, p=normalized)
        print(predicted)
        return predicted[0]



class PredictionsForText:

    def __init__(self, text: str):
        """Constructor"""
        self._text = text
        self._words = {}
        nlp = spacy.load("ru2")
        doc = nlp(text)
        for token in doc:
```

```
            self._words[token.text] = PredictionsForWord(token.text)

    def __str__(self):
        stroka = "-----------------------------------------"
        stroka += "Предсказания для текста: " + self._text + "\n"
        for word in self._words:
            stroka += str(word) + "\n"
        stroka += "-----------------------------------------"
        return stroka

    def add_bert(self, word: str, predicted: str, score, tokenizer: BertTokenizer):
        logger = logging.getLogger("graduateWork.PredictionsForText.add_bert")
        predictions = self._words.get(word)
        if predictions == None:
            logger.info("Word: %s not found in current text" % (word))
            return
        predictions.add_bert(predicted, score, tokenizer)

    def add_skipgram(self, word: str, predicted: str, score):
        logger = logging.getLogger("graduateWork.PredictionsForText.add_skipgram")
        predictions = self._words.get(word)
        if predictions == None:
            logger.info("Word: %s not found in current text" % (word))
            return
        predictions.add_skipgram(predicted, score)

    def add_fasttext(self, word: str, predicted: str, score):
        logger = logging.getLogger("graduateWork.PredictionsForText.add_fasttext")
        predictions = self._words.get(word)
        if predictions == None:
            logger.info("Word: %s not found in current text" % (word))
            return
        predictions.add_fasttext(predicted, score)

    def find_all(self, bert_tokenizer: BertTokenizer, udpipe: udpipe.Pipeline):
        for word in self._words:
            self._words[word].find_all(bert_tokenizer, udpipe)
```

```
def sinonimize(self):
    new_sentense = []
    for words in self._words:
        new_sentense.append(self._words[words].sinonimize())
    return ' '.join(new_sentense)

def sinonimize_skipgram(self):
    new_sentense = []
    for words in self._words:
        new_sentense.append(self._words[words].sinonimize_skipg())
    return ' '.join(new_sentense)

def sinonimize_fasttext(self):
    new_sentense = []
    for words in self._words:
        new_sentense.append(self._words[words].sinonimize_fast())
    return ' '.join(new_sentense)
```